DATA+AI SUMMIT BY databricks

LLMs in Enterprise

Nathan Azrak / Brian Law
June 11, 2024

AGENDA

LLMs in production: development to deployment, and common questions

Common issues with LLMs in enterprise

- How do we decide if finetuning is worth it?
- How do we serve such expensive models?
- How do we finetuning with sensitive training data?

Developing LLMs

- Evaluating your problem, when to use LLMs
- Training frameworks
- Training data construction and evaluation (case studies)
- The general developer flow

Data and training infrastructure

- Handling sensitive data for model training
- Exploration versus productionisation
- Cloud-agnostic training pipelines

LLM Training Challenges

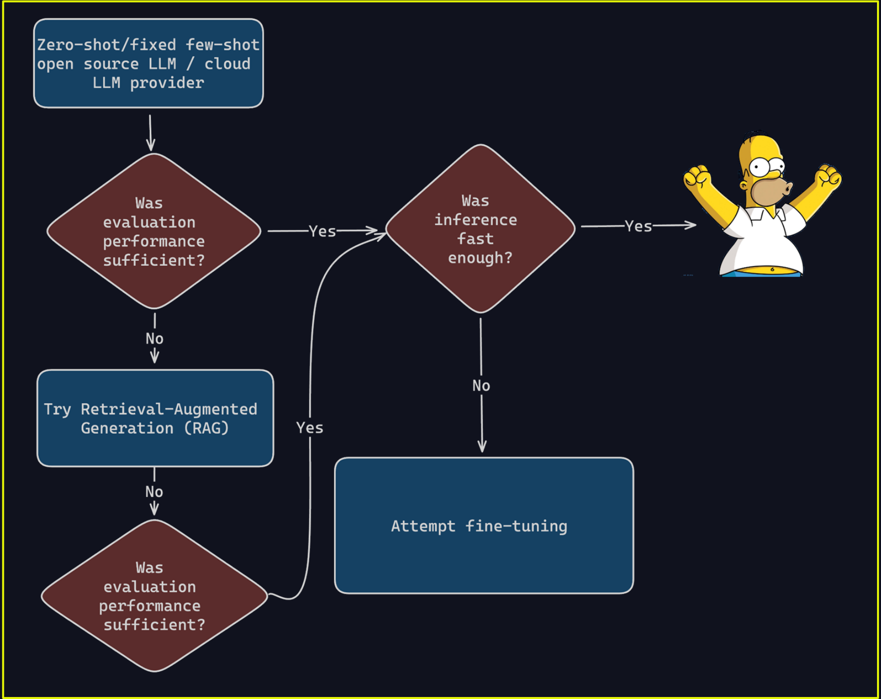
Training LLMs for production introduces many novel questions

- When do we decide to train an LLM?
  - High effort
  - How to justify?
- How do we construct useful training data?
  - How to create data for novel problems?
  - How to evaluate beyond CE Loss?
- How do we serve large, expensive models at scale?

Strategic Leveraging of LLMs

How to justify the finetuning of Large Language Models

- LLM finetuning is an uncertain process
- Effort is very high, with regards to:
  - Refining training code
  - Devising a dataset
  - Designing an evaluation method
- Finetuning should generally be a last resort



Model finetuning

Guidelines for choosing a base model

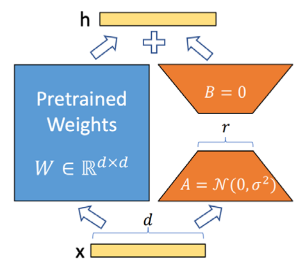
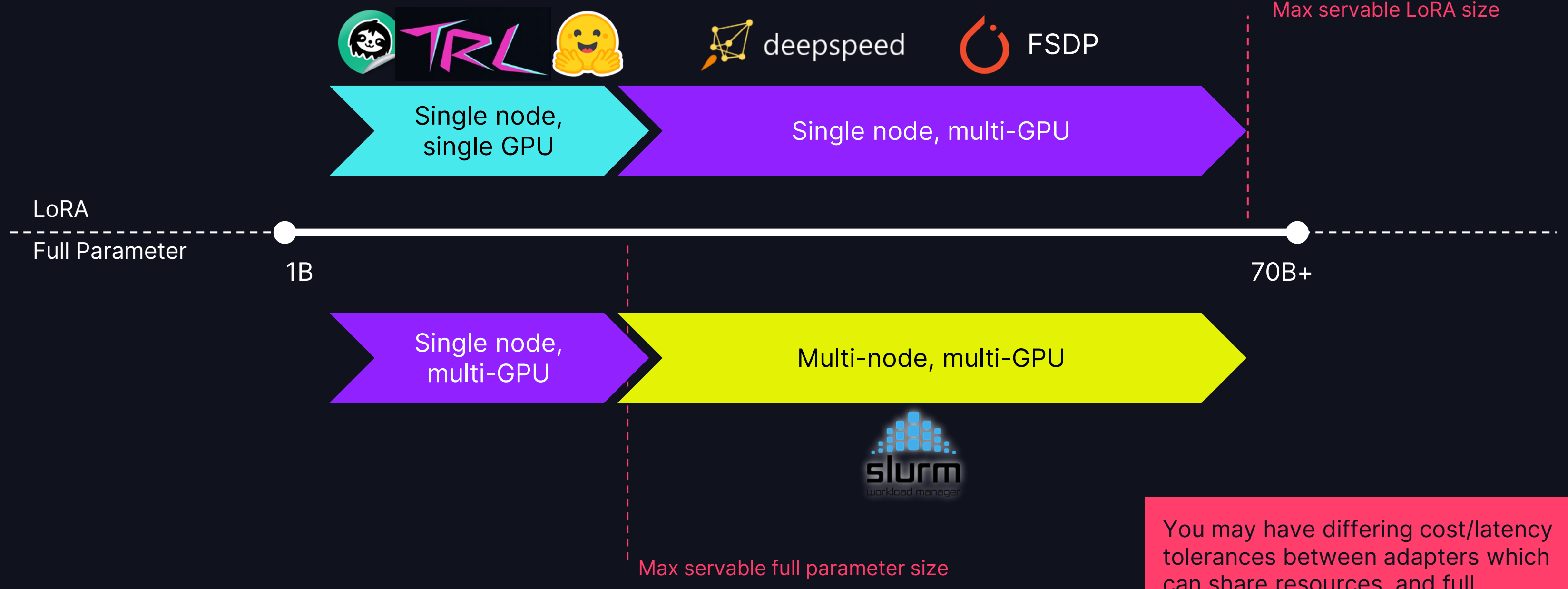


Figure 1: Our reparametrization. We only train A and B.

- Prioritise pragmatism, use trainers
- Start small and increase
- Prefer LoRA over full parameter
- Easier/cheaper to train
- Easier to deploy cheaply and efficiently
- Big unlock for LLM-driven features with low volume

Spectrum of LLM training frameworks

The tooling used depends on the size of the finetuning job



You may have differing cost/latency tolerances between adapters which can share resources, and full finetunes which require their own

Model finetuning

Zeroing in on a base model

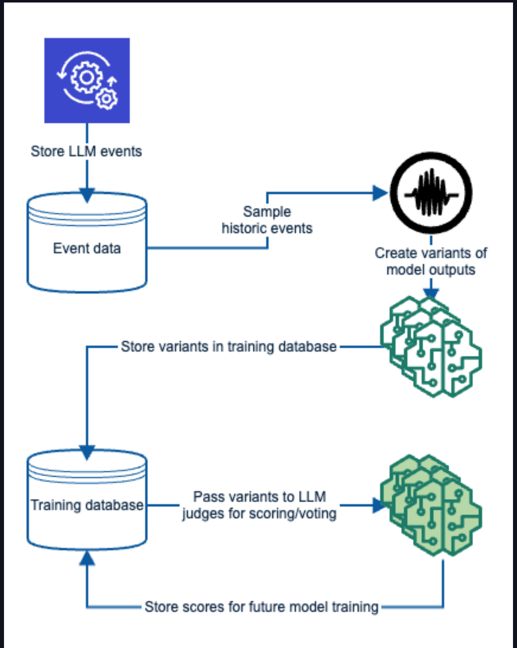
Ability to overfit indicates sufficient complexity in training parameters to learn the problem

- Test full parameter training if it is viable, continue if:
  - Significant gap in performance
  - feature is very high value or very high volume
- Fix hyperparameters for final candidate models, then iterate on training + eval sets

Data Generation

Continuous improvement through automatic dataset generation

- Dogfood prototype feature, use user queries to generate variants and training data
- LLM judges score results
  - Can create preference datasets, or use best results for SFT
- Can use human judges in future to create a subset
  - Metamodel on LLM judges to estimate score calibrated with human preference



Case Study: Code Generation

Code generation case study: overview

Generate training data

Creating useful demonstrative examples of the language

Model training

Training a variety of models from 7B to 34B, LoRA and full param

Offline evaluation

Evaluating using offline metrics, and summary analysis

Deploy internally

Dogfooding to Atlassian users to test quality with real users

Review customer feedback

Manually evaluate written feedback

Case Study: Code Generation

Code generation case study: data generation process

Extract HTML docs

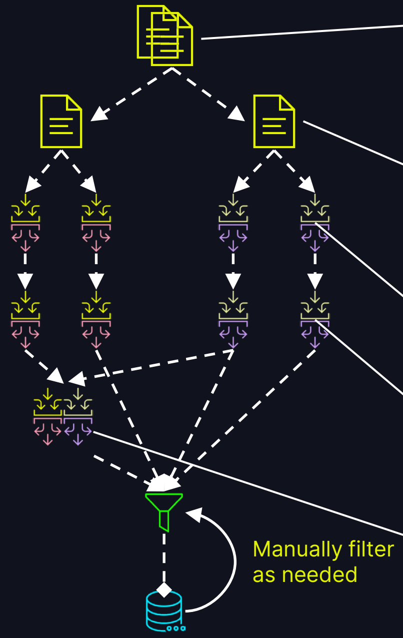
Process into json descriptors

Generate input/output pairs, with user queries (LLM)

Paraphrase and create variants (LLM)

Combine inputs and outputs to increase complexity (LLM)

Perform automated filter to remove ambiguous queries (LLM)




```
<!DOCTYPE html><html><head><title>Function
Documentation</title></head><body><h1>Power
Function</h1><p><b>def</b> pow(a: float, pow: float) ->
float</p><p>Returns <code>a</code> to the power
<code>pow</code>.</p><h2>Example:</h2><pre>result = pow(2, 3)
# Expected: 8</pre></body></html>
```

```
{
  "name": "pow",
  "inputs": [
    {"name": "a", "type": "float"},
    {"name": "pow", "type": "float"}
  ],
  "output": {"type": "float"},
  "description": "Returns `a` to the power `pow`.",
  "examples": [{"code": "result = pow(2, 3)", "expected": "8"}
  ]
}
```

```
{"input": "What is 2 to the power of 3?", "output": "pow(2.,
3.)"}
```

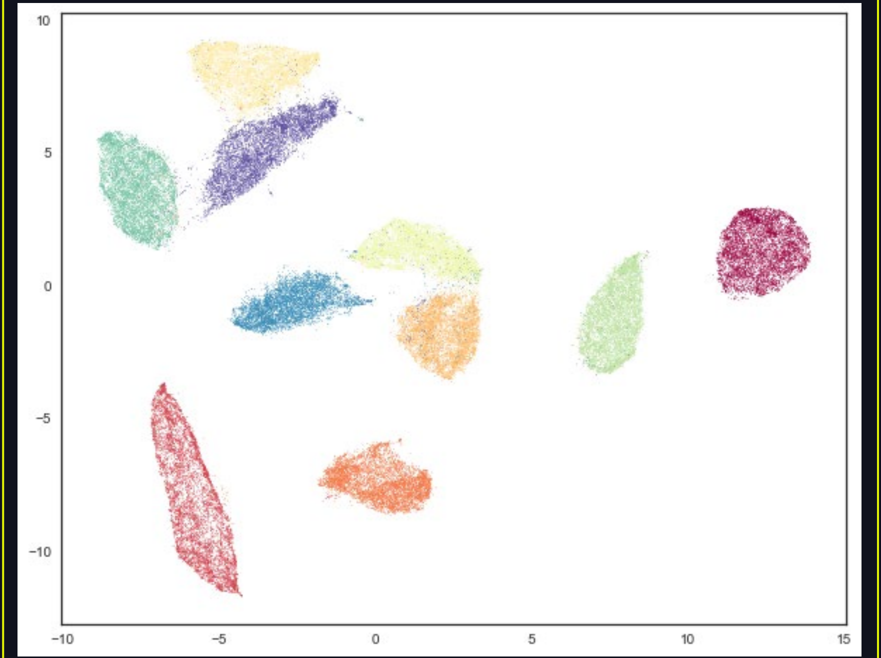
```
"What is 2 to the power of 3?" -> "what's 2 cubed?"
```

```
{"input": "What is to to the power of 3 plus 20?", "output":
"add(pow(2., 3.), 20.)"}
```

Case Study: Code Generation

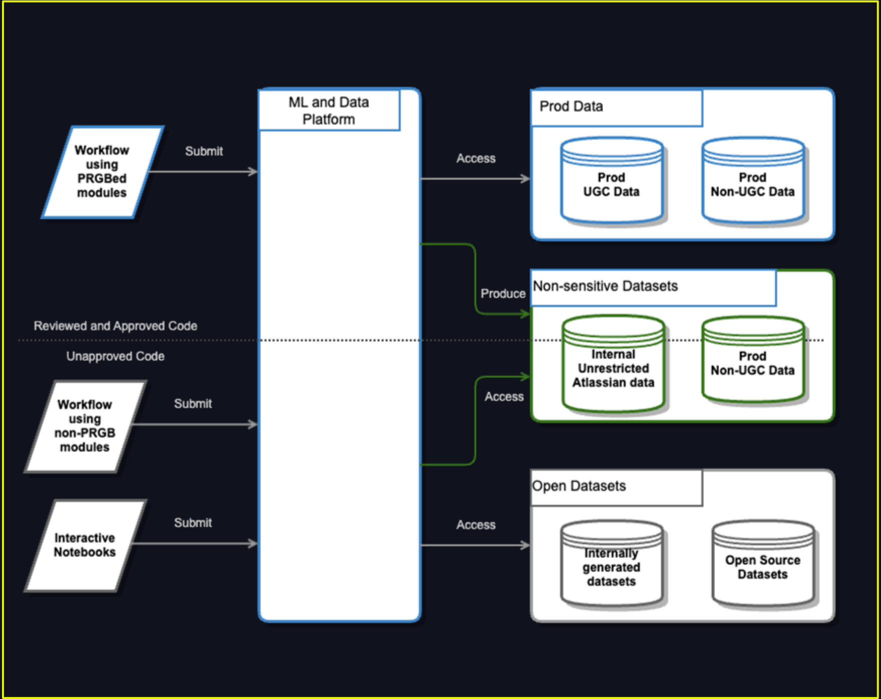
Code generation case study: evaluating fail cases

- Analysed failures by:
  - Clustering inputs from failures
  - Clustering model outputs from failures
  - Manually analysing clusters for themes
- Restart loop, generating more examples to counter failure cases
- Solving impossible queries
  - Simple classifier or basic LLM call to filter non-English or impossible queries
  - Catch these before querying LLM



ML Training Platform

Building a platform for safe, convenient ML development



- Platform built to secure customer data while not prohibiting rapid development
- Yaml-driven workflows assist with reproducibility
- IAM and access controls are used to limit access to sensitive data

Insights

Insights for model building and platform development

- Model training takes ages and is a bit of a pain! Make it easy on yourself
- Make data generation code generic and extendable - your first dataset will NOT be your last dataset
- Encourage version control and config-driven code
- Utilise an experiment tracker, and track package versions as well as run configurations for reproducibility

PRACTICAL FINETUNING

ANSWER THESE QUESTIONS FIRST

Quick spot check before you start!

- Have you tried advanced prompt engineering?
- Do you have samples of required inputs / outputs?
- Do you need very specific bespoke logic?

THE VARIOUS TYPES OF TRAINING

When to use

LoRa/QLoRa

- Most VRAM Efficient
- Not as performant if adding new domain knowledge

Full Parameter Finetune

- Better performance than LoRa
- Requires more time and GPU compute
- Higher chance of forgetting

Continued Pretrain

- Expensive and requires a lot of data and gpu compute
- Critical for learning new:
    - Languages
    - Domains
- Chance of forgetting

THE VARIOUS TYPES OF TRAINING

When to use

LoRa/QLoRa

- Most VRAM Efficient
- Not as performant if adding new domain knowledge

Full Parameter Finetune

- Better performance than LoRa
- Requires more time and GPU compute
- Higher chance of forgetting

Continued Pretrain

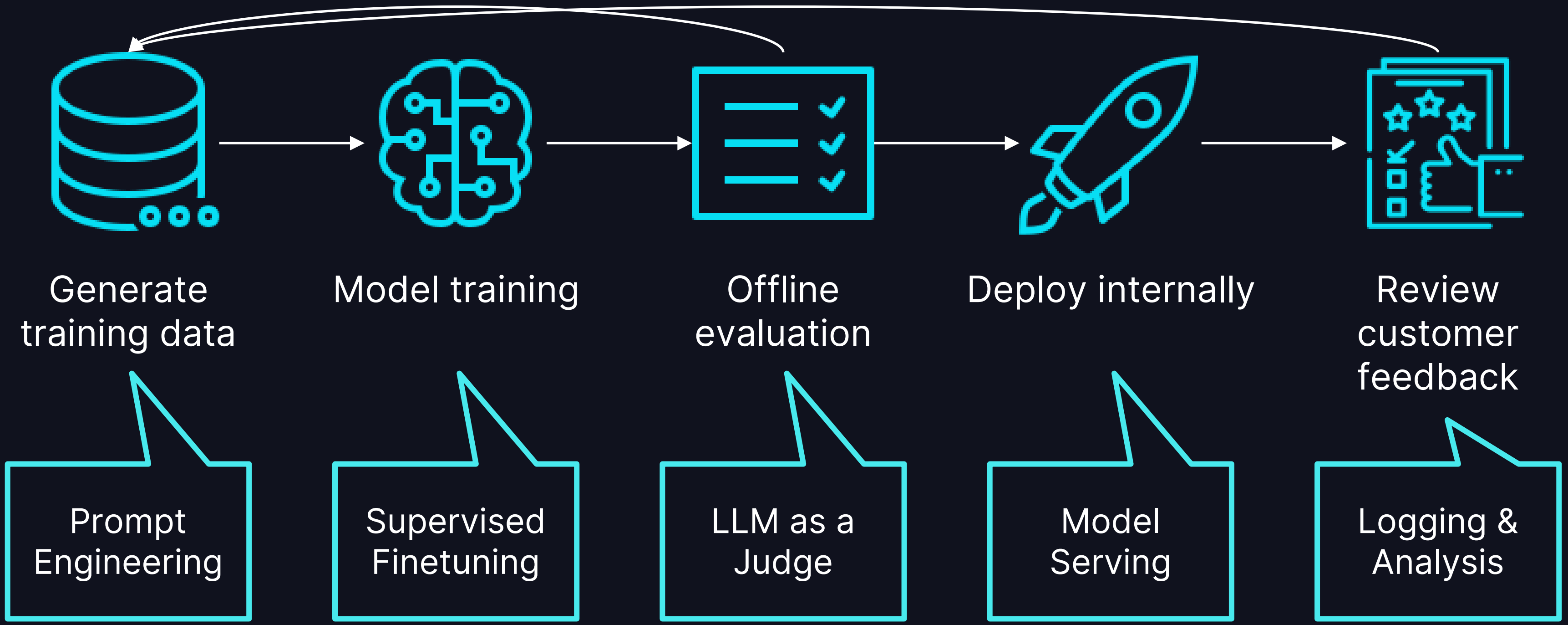
- Expensive and requires a lot of data and gpu compute
- Critical for learning new:
  - Languages
  - Domains
- Chance of forgetting

The Finetuning Loop

Rehash from before

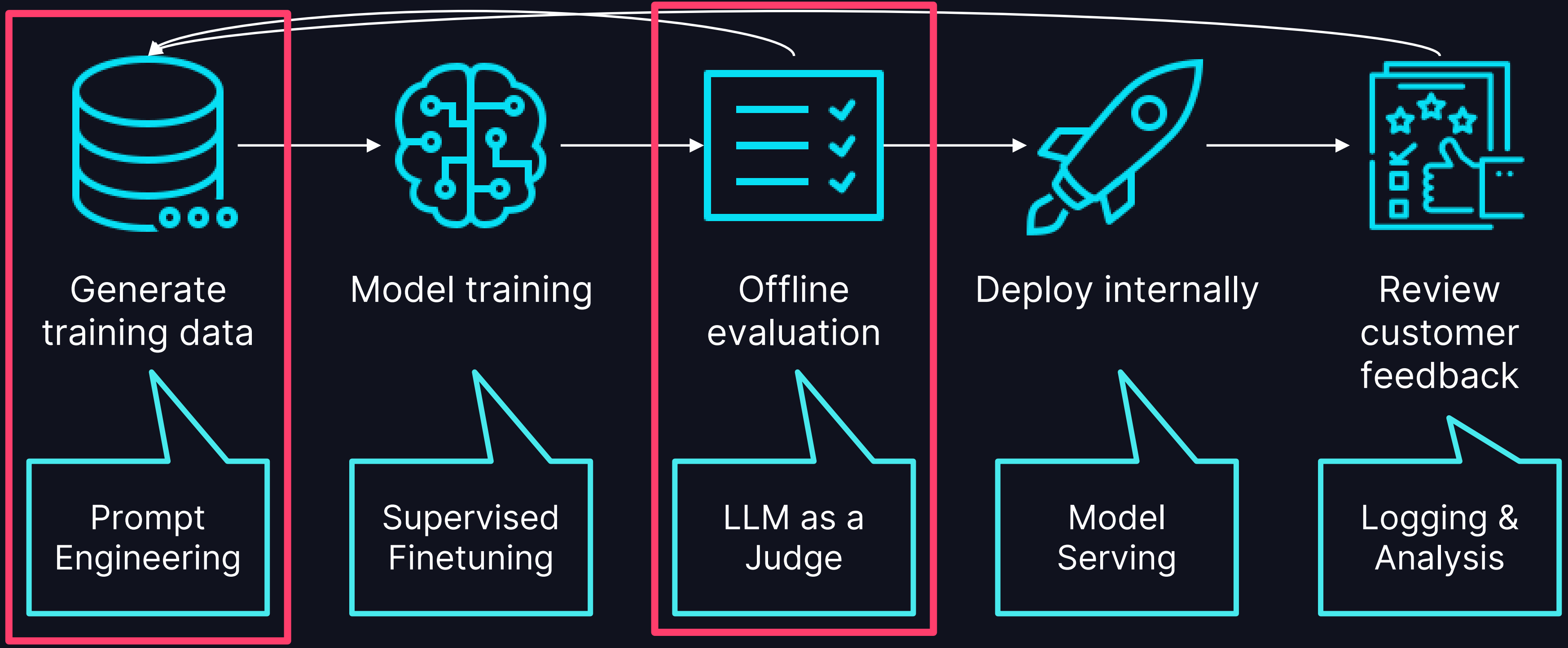
The Finetuning Loop

Key Techniques



The Finetuning Loop

The dataset and evaluations are key

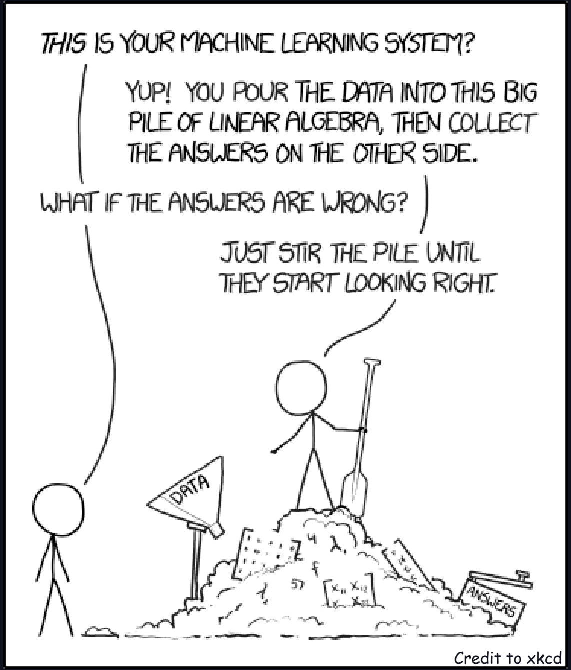


SETTING UP YOUR DATA

Sorting out your dataset

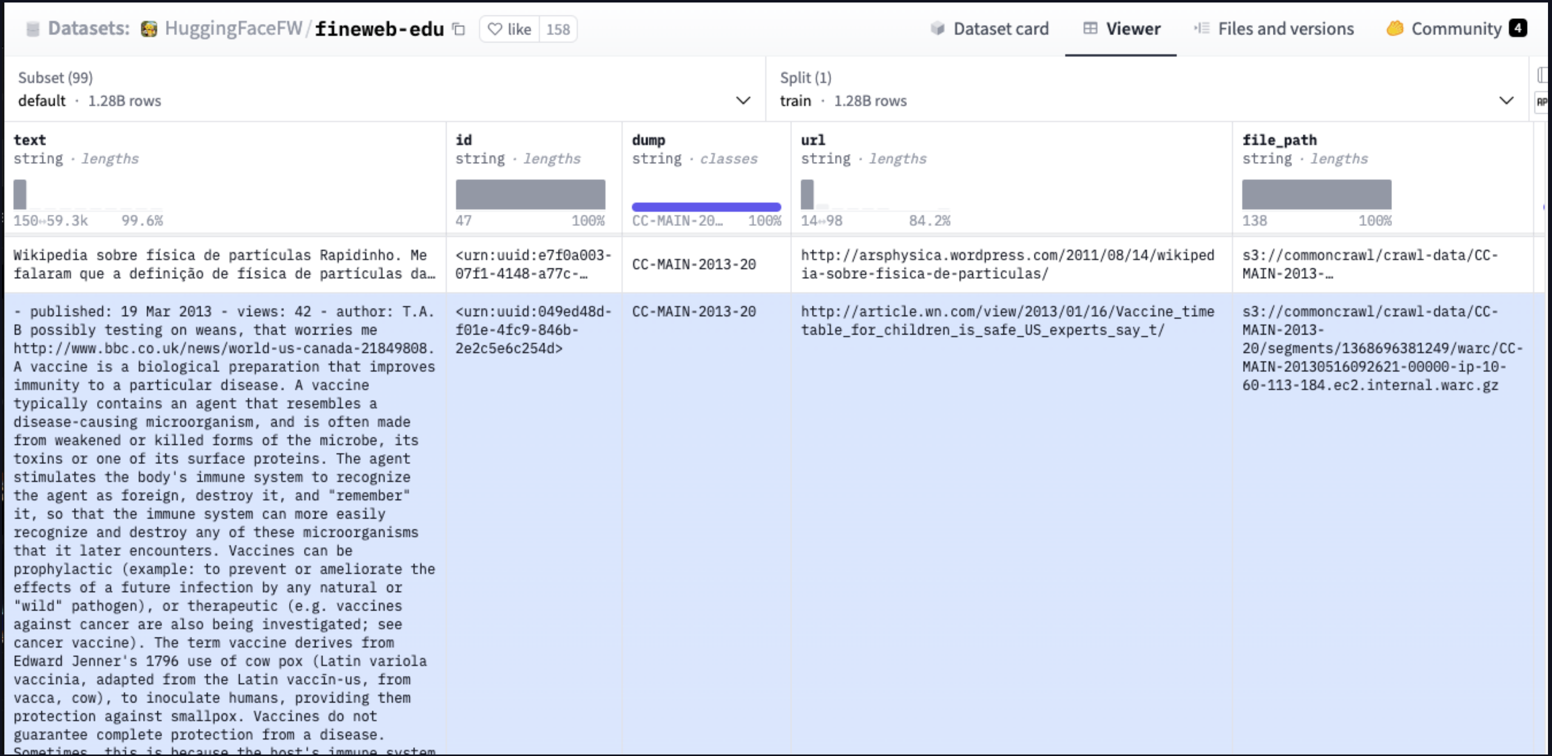
Common Questions

- How much?
- How to format?
- How to build?



TYPES OF DATA

Raw Text - (https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu/viewer/default/train?row=16)



Datasets: HuggingFaceFW/fineweb-edu | like 158 | Dataset card | Viewer | Files and versions | Community 4

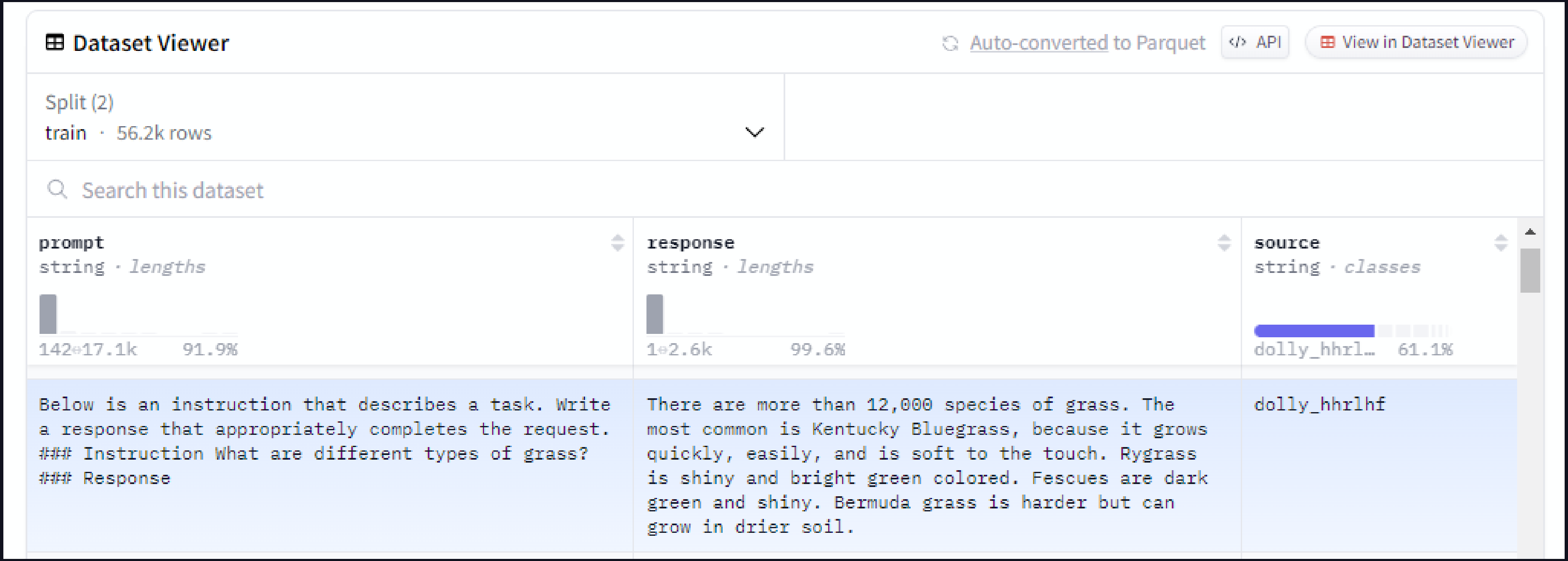
Subset (99): default · 1.28B rows

Split (1): train · 1.28B rows

text	id	dump	url	file_path
string · lengths 150↔59.3k 99.6%	string · lengths 47 100%	string · classes CC-MAIN-20… 100%	string · lengths 14↔98 84.2%	string · lengths 138 100%
Wikipedia sobre física de partículas Rapidinho. Me falaram que a definição de física de partículas da…	<urn:uuid:e7f0a003-07f1-4148-a77c-…	CC-MAIN-2013-20	http://arsphysica.wordpress.com/2011/08/14/wikipedia-sobre-fisica-de-particulas/	s3://commoncrawl/crawl-data/CC-MAIN-2013-…
- published: 19 Mar 2013 - views: 42 - author: T.A. B possibly testing on weans, that worries me http://www.bbc.co.uk/news/world-us-canada-21849808. A vaccine is a biological preparation that improves immunity to a particular disease. A vaccine typically contains an agent that resembles a disease-causing microorganism, and is often made from weakened or killed forms of the microbe, its toxins or one of its surface proteins. The agent stimulates the body's immune system to recognize the agent as foreign, destroy it, and "remember" it, so that the immune system can more easily recognize and destroy any of these microorganisms that it later encounters. Vaccines can be prophylactic (example: to prevent or ameliorate the effects of a future infection by any natural or "wild" pathogen), or therapeutic (e.g. vaccines against cancer are also being investigated; see cancer vaccine). The term vaccine derives from Edward Jenner's 1796 use of cow pox (Latin variola vaccinia, adapted from the Latin vaccīn-us, from vacca, cow), to inoculate humans, providing them protection against smallpox. Vaccines do not guarantee complete protection from a disease. Sometimes, this is because the host's immune system	<urn:uuid:049ed48d-f01e-4fc9-846b-2e2c5e6c254d>	CC-MAIN-2013-20	http://article.wn.com/view/2013/01/16/Vaccine_timetable_for_children_is_safe_US_experts_say_t/	s3://commoncrawl/crawl-data/CC-MAIN-2013-20/segments/1368696381249/warc/CC-MAIN-20130516092621-00000-ip-10-60-113-184.ec2.internal.warc.gz

TYPES OF DATA

Prompt - Response Data - (https://huggingface.co/datasets/mosaicml/instruct-v3?row=1)



TYPES OF DATA

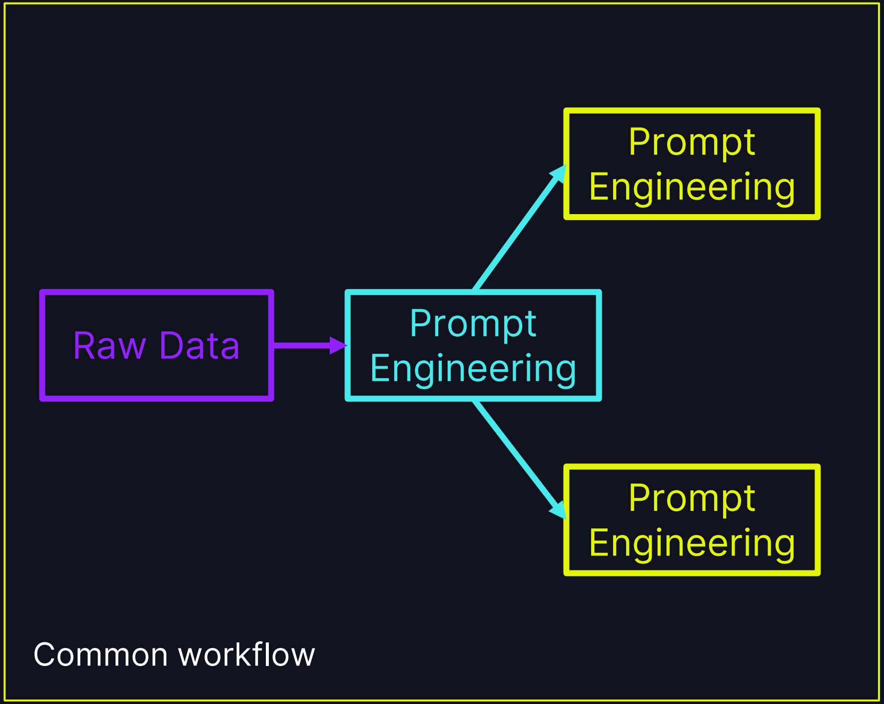
Chat Dataset - (https://huggingface.co/datasets/erfanzar/ShareGPT4?row=5)


```
[
  {
    "role": "user",
    "value": "act as an expert pharmacologist. explain to a physician the potential interactions of the medications named here:\n\n Wellbutrin XL to 150 mg daily for atte
  },
  {
    "role": "assistant",
    "value": "As a pharmacologist, I would like to bring to your attention some potential drug interactions and considerations among the medications listed for this patie
  },
  { "role": "user", "value": "include gabapentin" },
  {
    "role": "assistant",
    "value": "Adding gabapentin to the patient's medication regimen introduces the following considerations and potential interactions:\n\n1. Gabapentin and Cymbalta (dul
  },
  {
    "role": "user",
    "value": "can wellbutrin contribute to the risk of seratonin syndrome"
  },
  {
    "role": "assistant",
    "value": "I apologize for the confusion in my previous response. While Wellbutrin (bupropion) is not typically associated with an increased risk of serotonin syndrome
  },
  {
    "role": "user",
    "value": "which of these medications contribute to seratonin syndrome risk"
  },
  {
    "role": "assistant",
    "value": "I apologize for any confusion. Of the medications listed for the patient, Cymbalta (duloxetine) is the primary one associated with an increased risk of sero
  },
  { "role": "user", "value": "doesn't trazodone?" },
  {
    "role": "assistant",
    "value": "You are correct, and I apologize for the oversight. Trazodone is indeed another medication in the patient's list that can contribute to the risk of serotoni
  }
]
```

HOW CAN WE MAKE DATA?

Data needs to be representative

- Customer Q&A Pages
- Call Centre Logs
- Hand Crafted
- Synthetically Generated



GENERATING SYNTHETIC DATA

Most will do this

We can use LLMs:

- To create sample questions and associated answers

- To create synthetic examples of a conversation

```
seed_question_prompt = Prompt(
    name="seed_question",
    instruction="Generate a question that can be fully answered from given context. The question should be formed using topic",
    examples=[
        {
            "context": "Photosynthesis in plants involves converting light energy into chemical energy, using chlorophyll and ot
            "keyphrase": "Photosynthesis",
            "question": "What is the role of photosynthesis in plant growth?",
        },
        {
            "context": "The Industrial Revolution, starting in the 18th century, marked a major turning point in history as it l
            "keyphrase": "Industrial Revolution",
            "question": "How did the Industrial Revolution mark a major turning point in history?",
        },
        {
            "context": "The process of evaporation plays a crucial role in the water cycle, converting water from liquid to vapo
            "keyphrase": "Evaporation",
            "question": "Why is evaporation important in the water cycle?",
        },
    ],
    input_keys=["context", "keyphrase"],
    output_key="question",
    output_type="str",
)
```

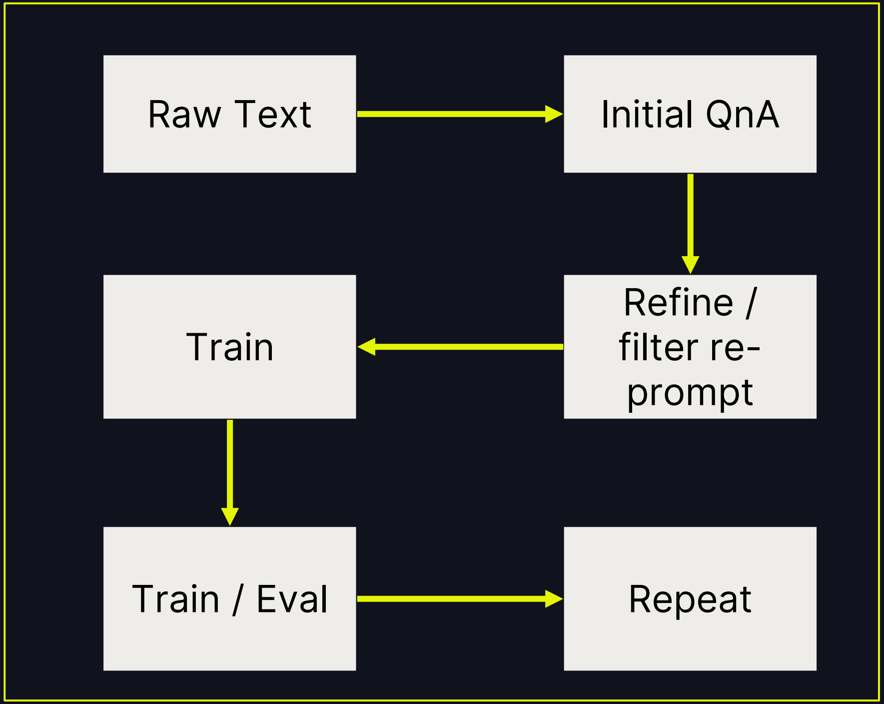
source: ragas source code (https://github.com/explodinggradients/ragas/blob/main/src/ragas/testset/prompts.py)

GENERATING SYNTHETIC DATA

The full workflow

Consider also:

- Answer Length
- Tone and syntax
- Thoroughness of answer
- Quality of the dataset



HOW MUCH DATA IS ENOUGH?

YOU CAN START WITH x00 BUT THE MORE THE BETTER

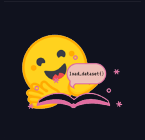
ONTO THE TRAINING LOOP

Input Dataset Formatting and preprocessing

Main Train Function

Logging and Monitoring

JSONL



streaming

ONTO THE TRAINING LOOP

Input Dataset Formatting and preprocessing

Main Train Function

Logging and Monitoring

TRL

FSDP

deepspeed

composer

ONTO THE TRAINING LOOP

Input Dataset Formatting and preprocessing

Main Train Function

Logging and Monitoring

mlflow

Weights & Biases

TRAINING TECHNIQUES

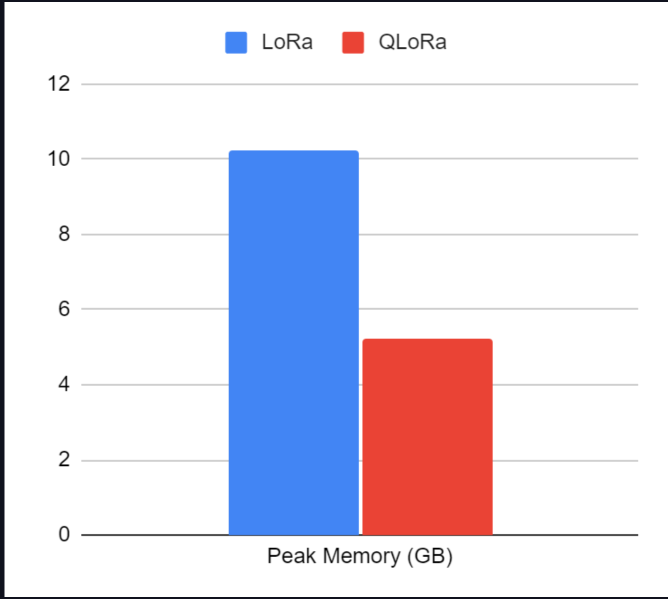
LET'S UNDERSTAND VRAM - (https://github.com/AnswerDotAI/fsdp_qlora/blob/main/benchmarks_03_2024.md)

Machine:

- 2x 24GB VRAM consumer cards
- 128GB CPU RAM

Model:

- Llama 2 - 7B
- 2048 Context Length
- Batch Size 1



TRAINING TECHNIQUES

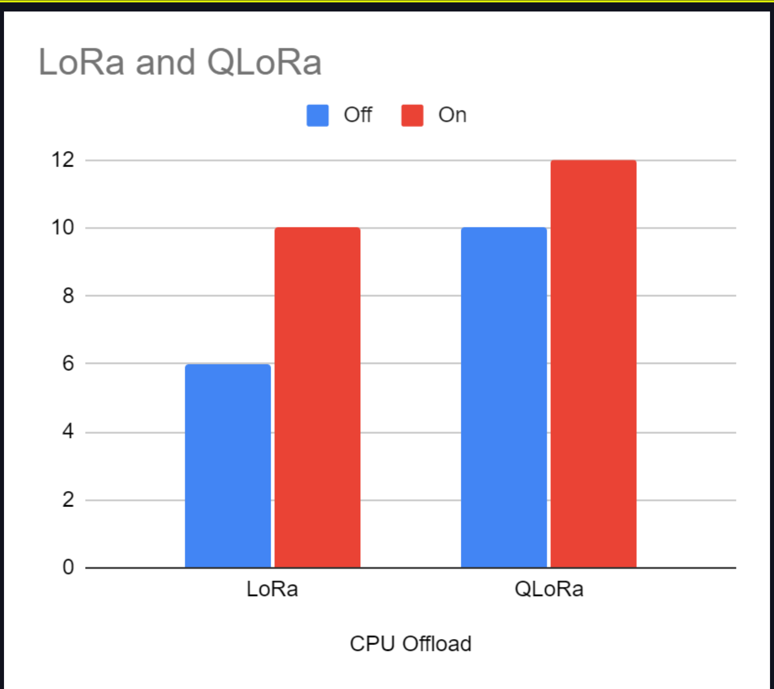
LET'S UNDERSTAND VRAM - (https://github.com/AnswerDotAI/fsdp_qlora/blob/main/benchmarks_03_2024.md)

Machine:

- 2x 24GB VRAM consumer cards
- 128GB CPU RAM

Model:

- Llama 2 - 7B
- 2048 Context Length



TRAINING TECHNIQUES

Navigating VRAM

- Make sure model fits
- Use LoRa
  - Can test QLoRa - check evals!
- Increment batch size till out of RAM
  - (Optional CPU Offload)

UNDERSTANDING DATA PREPROCESSING

STEP 1 - Raw Data

- Language is complex
- Sentences have different lengths
- This is not good for nice matrix multiplications

UNDERSTANDING DATA PREPROCESSING

STEP 2 - Encode

- 128000, 14126, 374, 6485
- 128000, 32458, 2436, 617, 2204, 29416
- 128000, 2028, 374, 539, 1695, 369, 6555, 6303, 12842, 10939

UNDERSTANDING DATA PREPROCESSING

STEP 3 - Pad

- 128000, 14126, 374, 6485, 0, 0, 0, 0, 0, 0, 0, 0
- 128000, 32458, 2436, 617, 2204, 29416, 0, 0, 0, 0, 0, 0
- 128000, 2028, 374, 539, 1 695, 369, 6555, 6303, 12842, 10939, 0, 0

UNDERSTANDING DATA PREPROCESSING

STEP 4 - Shard

- 128000, 14126, 374, 6485, 0, 0, 0, 0, 0, 0, 0, 0

GPU1

- 128000, 32458, 2436, 617, 2204, 29416, 0, 0, 0, 0, 0, 0

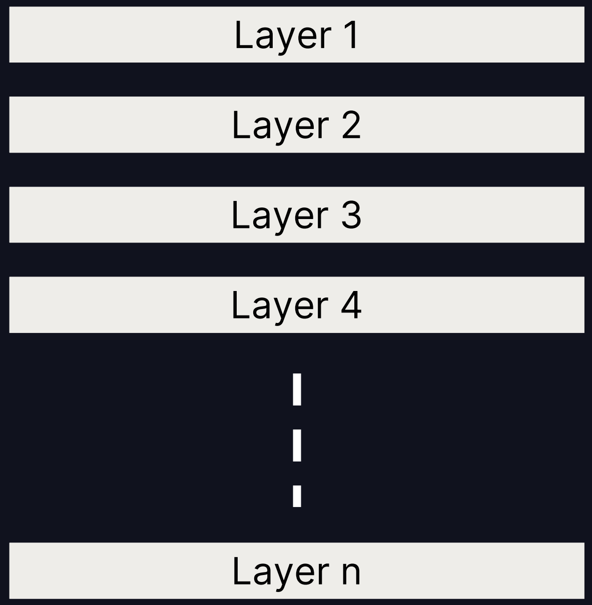
GPU2

- 128000, 2028, 374, 539, 1 695, 369, 6555, 6303, 12842, 10939, 0, 0

GPU3

UNDERSTANDING MAIN TRAINING LOOP

1) Load Model



- Load Weights from file
- Adjust Layers for quantization / LoRa as needed
- Move and Shard to GPU

UNDERSTANDING MAIN TRAINING LOOP

2) Configure / Load Rest of Parameters

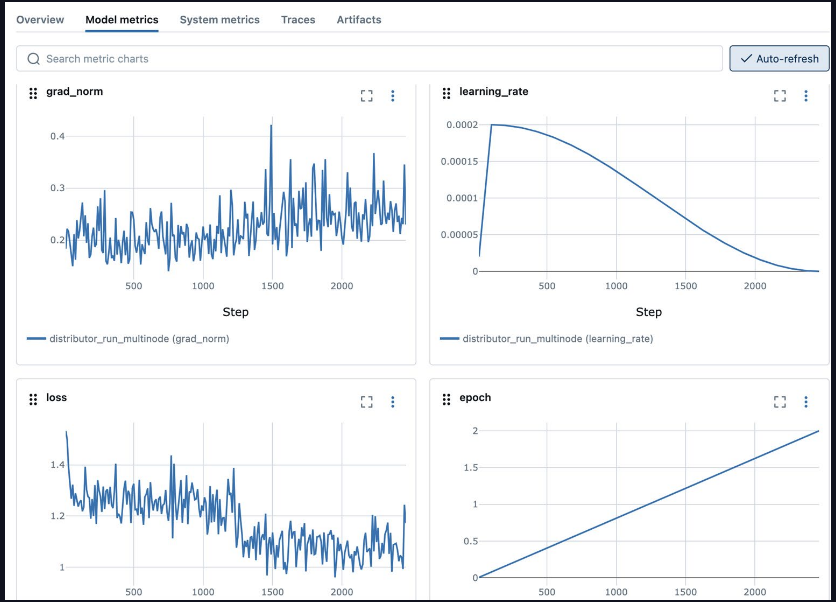
```
def train_one_epoch(epoch_index, tb_writer):
    running_loss = 0.
    last_loss = 0.

    # Here, we use enumerate(training_loader) instead of
    # iter(training_loader) so that we can track the batch
    # index and do some intra-epoch reporting
    for i, data in enumerate(training_loader):
        # Every data instance is an input + label pair
        inputs, labels = data

        # Zero your gradients for every batch!
        optimizer.zero_grad()

        # Make predictions for this batch
        outputs = model(inputs)

        # Compute the loss and its gradients
        loss = loss_fn(outputs, labels)
        loss.backward()

        # Adjust learning weights
        optimizer.step()

        # Gather data and report
        running_loss += loss.item()
        if i % 1000 == 999:
            last_loss = running_loss / 1000 # loss per batch
            print('  batch {} loss: {}'.format(i + 1, last_loss))
            tb_x = epoch_index * len(training_loader) + i + 1
            tb_writer.add_scalar('Loss/train', last_loss, tb_x)
            running_loss = 0.

    return last_loss
```

Components to setup and configure:

- Optimizer
- Learning Rate Schedule
- ZeRo configs / Sharding configs
- Loss Calculations

UNDERSTANDING MAIN TRAINING LOOP

3) Monitor



Training Can be unstable:

- Loss Spikes
- Checkpointing issues
- Hardware failure
- Loss calc mistakes

SCALING UP

How to speed up your training run:

Start with single GPU

Validate:

- Train loop runs
- Loss is decreasing as expected
- Checkpointing and logging working

Move to Double

Validate:

- Distribution is happening
- Loss is still decreasing as expected
- Logging and checkpointing works distributed

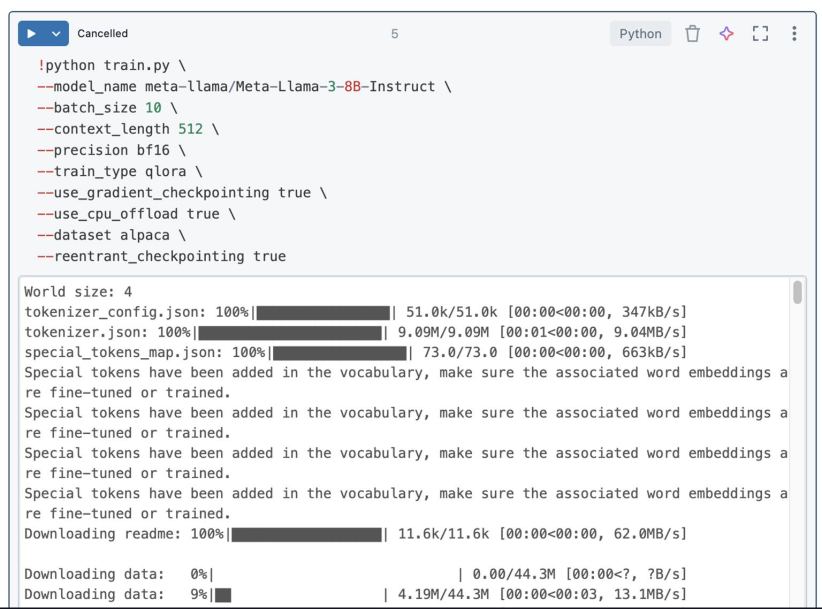
Max out on one node (x8 first)

After the previous tests, this should just work.

TRAINING ON DATABRICKS

Starting with Single Node

See: https://github.com/Data-drone/dais24_finetuning.git


Cancelled 5 Python

```
!python train.py \
--model_name meta-llama/Meta-Llama-3-8B-Instruct \
--batch_size 10 \
--context_length 512 \
--precision bf16 \
--train_type qlora \
--use_gradient_checkpointing true \
--use_cpu_offload true \
--dataset alpaca \
--reentrant_checkpointing true
```

```
World size: 4
tokenizer_config.json: 100%|██████████| 51.0k/51.0k [00:00<00:00, 347kB/s]
tokenizer.json: 100%|██████████| 9.09M/9.09M [00:01<00:00, 9.04MB/s]
special_tokens_map.json: 100%|██████████| 73.0/73.0 [00:00<00:00, 663kB/s]
Special tokens have been added in the vocabulary, make sure the associated word embeddings a
re fine-tuned or trained.
Special tokens have been added in the vocabulary, make sure the associated word embeddings a
re fine-tuned or trained.
Special tokens have been added in the vocabulary, make sure the associated word embeddings a
re fine-tuned or trained.
Special tokens have been added in the vocabulary, make sure the associated word embeddings a
re fine-tuned or trained.
Downloading readme: 100%|██████████| 11.6k/11.6k [00:00<00:00, 62.0MB/s]

Downloading data:   0%|          | 0.00/44.3M [00:00<?, ?B/s]
Downloading data:   9%|█         | 4.19M/44.3M [00:00<00:03, 13.1MB/s]
```

Use:

- Single Node MLR
- %sh or !python magics to execute code
- Init_script for os level dependencies

TRAIN LOOP ON DATABRICKS

Expand with TorchDistributor / Deepspeed Distributor

See: https://github.com/Data-drone/dais24_finetuning.git

```
if exec_opt == 'TorchDistributor':
    from pyspark.ml.torch.distributor import TorchDistributor

    def accelerate_train(mlflow_run_name:str='accelerate_run', deepspeed=None):

        peft_config, training_arguments = setup_params(shared_parameters=shared_parameters,
                                                       mlflow_run_name=mlflow_run_name)
        trainer = train(peft_config, training_arguments, dataset, distributor=True)

        return trainer

    num_gpus_per_node = num_gpus
    num_nodes = num_nodes
    num_processes = num_gpus_per_node * num_nodes
    local_status = True if num_nodes == 1 else False


    distributor = TorchDistributor(num_processes=num_processes,
                                   local_mode=local_status, use_gpu=True)

    logger.info(f"Launching job with TorchDistributor with {num_gpus_per_node} gpus per node and {num_nodes} nodes")
    completed_trainer = distributor.run(accelerate_train, f'distributor_run_multinode')

# COMMAND ----------
```

Node 1

GPU 1

GPU 2

Each GPU will have:
- Copy of weights
- Partial optimizer states
- Slice of data

Node 2

GPU 1

GPU 2

KEY SETTINGS & PARAMETERS

That you need to know

To get a better finetune

- Learning Rate

To make things fit on GPU

- ZeRo Stage
- Offload
- Batch Size / gradient accumulation
- Quantization

KEY SETTINGS & PARAMETERS

That you need to know

To get a better finetune

- Learning Rate
  - Explore these can be case specific

To make things fit on GPU

- Quantization
  - 16bit is a given 8 / 4 do some testing
- ZeRo Stage / Sharding
  - Use 3 generally / Full Shard
- Offload
  - Hardware dependent
- Batch Size / gradient accumulation
  - Adjust to make best use of VRAM but leave a little buffer
  - Use gradient accumulation if targeting specific batch size

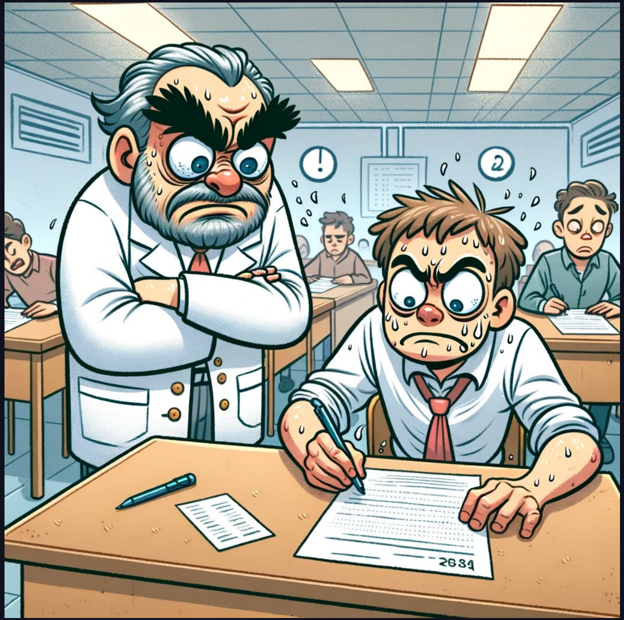
RUNNING EVALS

Accessing Success

- Have representative questions

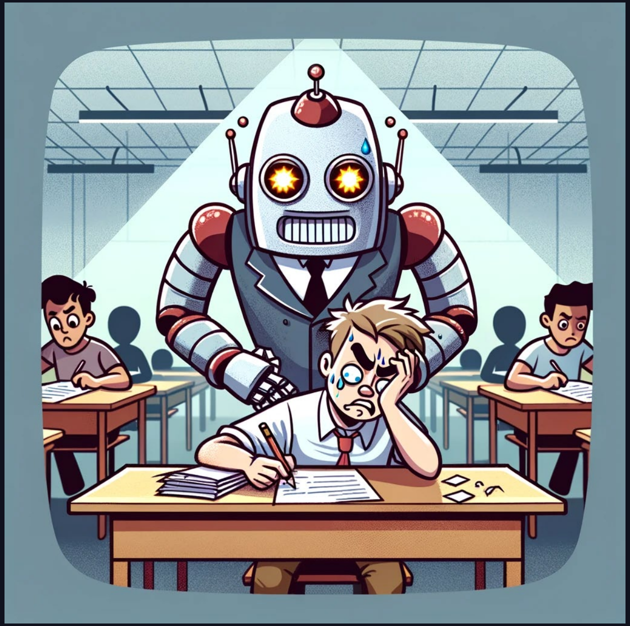
- Test general knowledge too in case of forgetting

- Make sure to test many topics and question types



Typical Evals

Use LLM-as-a-Judge

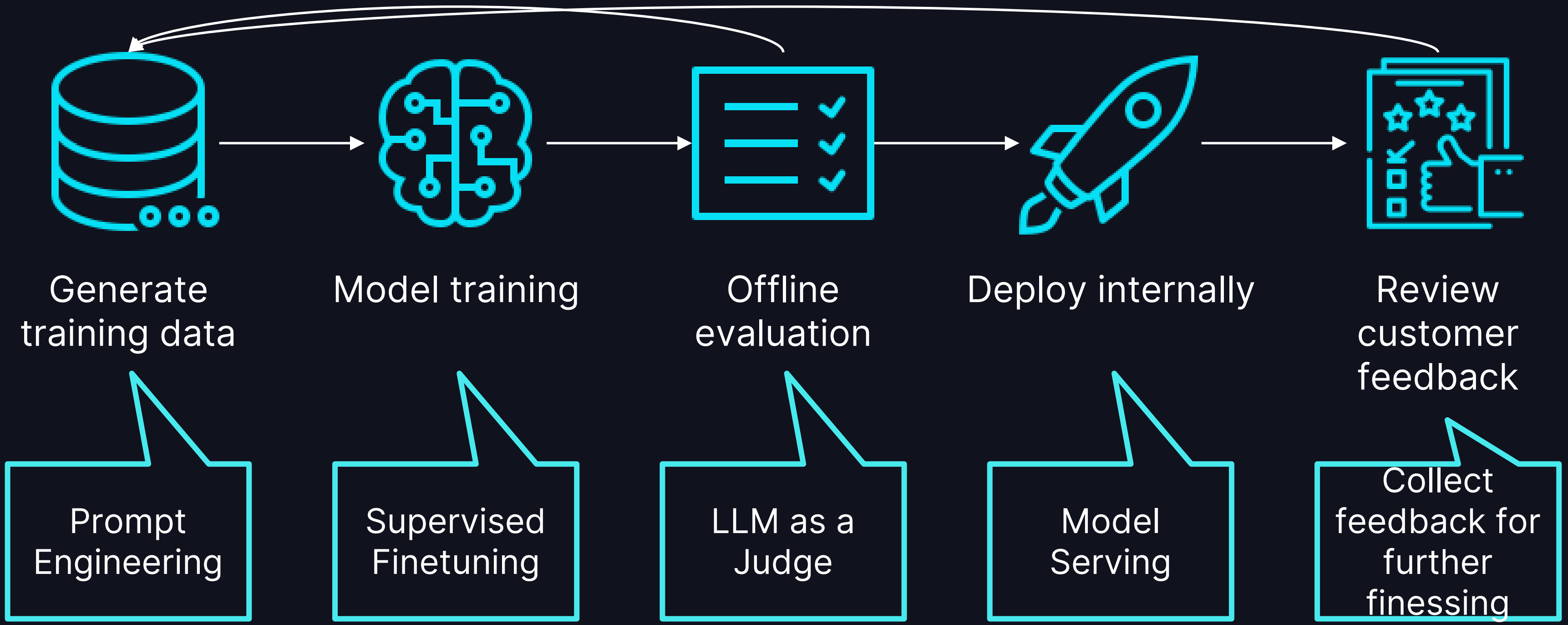


To Scale:

- It is common to use LLM to judge response
- LLM-as-a-Judge =/= customer preferences!
  - Be sure to calibrate
- Manual Work will be required

The Finetuning Loop

A quick review



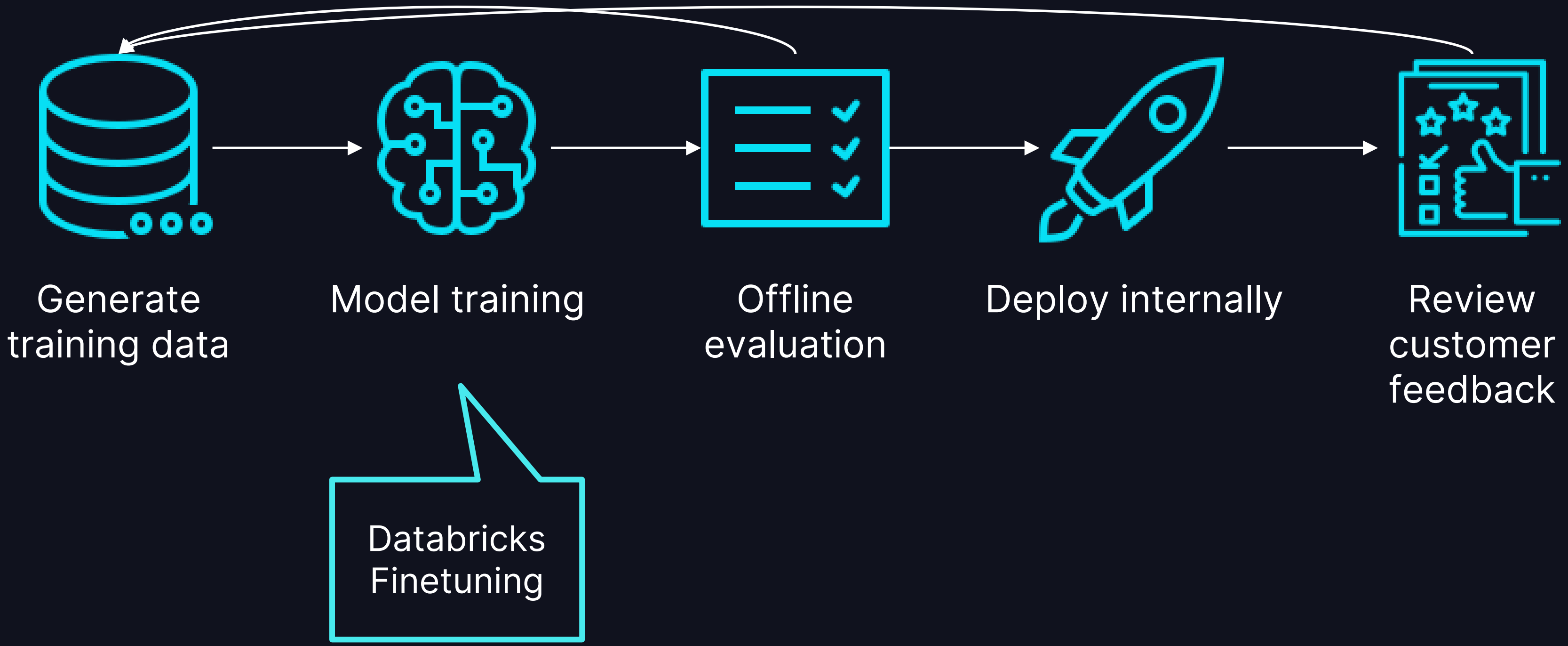
One Last thing

Coming soon!

Databricks Finetuning

The Finetuning Loop

A quick review



Databricks Finetuning in a minute

Full SaaS Solution

- No GPU worries
- No scaling worries
- No Boilerplate Training Loop

One Simple API so that you can focus on Data and Evals

Python

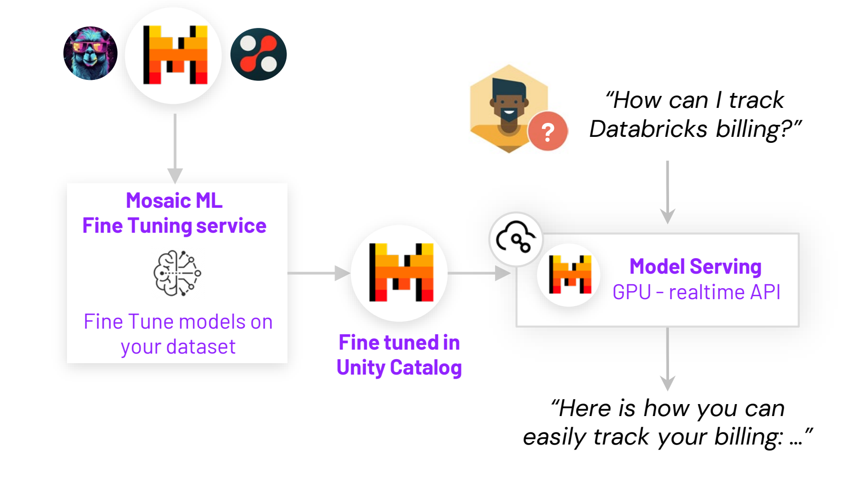
```python
from databricks.model_training import foundation_model as fm

run = fm.create(
    model="databricks/dbrx-base",
    train_data_path="dbfs:/Volumes/main/mydirectory",
    register_to="main.mydirectory.myname"
    training_duration="1ep",
    learning_rate="5e-7",
)
```

Try Mosaic AI & LLM Fine Tuning now!

SCAN ME

[Open the demo page]

Fine tune OSS models with your dataset




```
dbdemos.install('llm-fine-tuning')
```

Currently US Regions Only