LLMs in Production: Learning from Experience

M Waleed Kadous
Chief Scientist, Anyscale

AI Conference
Sept 26, 2023

anyscale

Key Takeaways

- LLMs: easy to demo, hard to productionize
- [Poll] Learned a few lessons we want to share
- Challenges & overcoming them
  - Cost
  - Data, Privacy
  - Deployment complexity
- What does the future hold so we can prepare?
  - Applications are not a single LLM call
  - RAG will be the default way enterprises use LLMs
  - Fine tuning for cost reduction

Who is Anyscale? Why should you listen to us?

- Company behind the Open Source project Ray
- Widely used Scalable AI Platform used by many companies
- What scalable means:
  - **Distributed**: Up to 4,000 nodes, 16,000 GPUs
  - **Efficient:** Keep costs down by efficient resource mgmt
  - **Reliable**: Fault tolerant, highly available
- Widely used by GenAI companies e.g. OpenAI, Cohere
- ChatGPT trained using Ray

Adobe Pinterest LinkedIn NIANTIC

instacart Spotify® DOORDASH

cohere Uber OpenAI samsara

ByteDance Uber Eats NETFLIX

airbnb RIOT GAMES® ANT GROUP

Why Ray?

instacart

12x

faster

samsara

50%

cheaper

Pinterest

40%

cheaper

amazon

90%

cheaper

Clari

5x

faster

DOORDASH

30%

cheaper

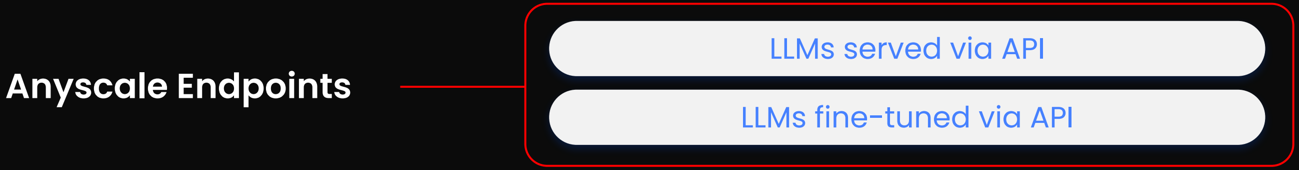
What’s our experience with LLMs?

We *provide* LLMs as a service (Llama models)

We *use* LLMs to make our products better

We *help* our customers deploy LLMs on Ray and on the managed version of Ray (Anyscale Platform)

Anyscale Endpoints



Anyscale Endpoints

LLMs served via API

LLMs fine-tuned via API

endpoints.anyscale.com

LLM Serving	**Price** **(per million tokens)**
Llama2 70B Codellama 34B	**$1.00**
Llama2 13B	**$0.25**
Llama2 7B	**$0.15**

Anyscale Endpoints

Cost efficiency touches every layer of the stack

- Single GPU optimizations
- Multi-GPU modeling
- Inference server
- Autoscaling
- Multi-region, multi-cloud

Anyscale Endpoints

$1 / million

tokens

(Llama-2 70B)

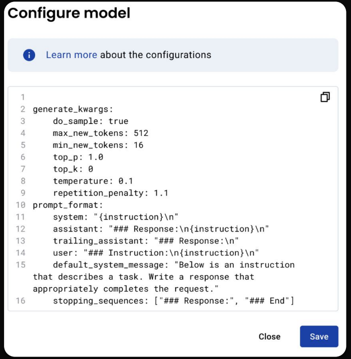
End-to-end LLM privacy, customization and control

Anyscale Endpoints

- LLMs served via API
- LLMs fine-tuned via API

Anyscale Private Endpoints

- Serve your LLMs from your Cloud
- Fine-tune & customize in your Cloud



Configure model

Learn more about the configurations

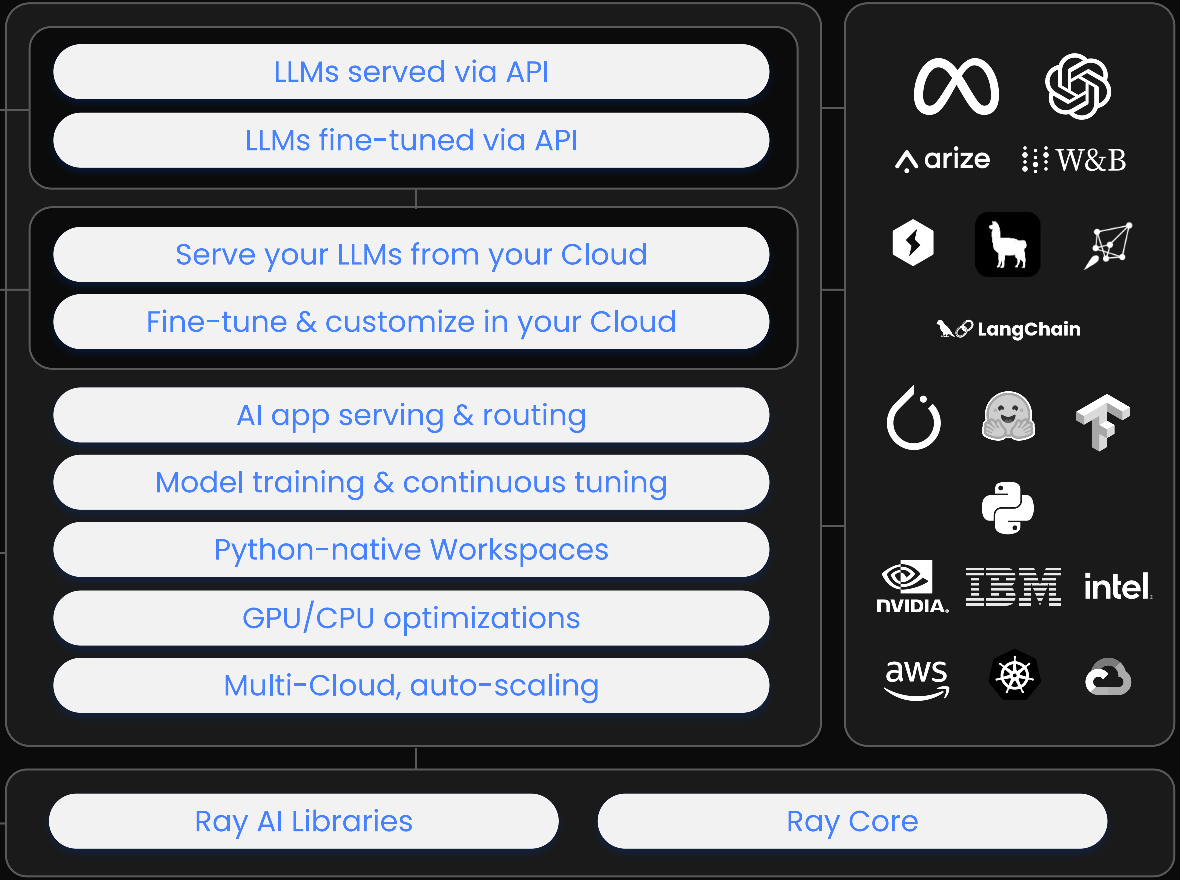
```
generate_kwargs:
    do_sample: true
    max_new_tokens: 512
    min_new_tokens: 16
    top_p: 1.0
    top_k: 0
    temperature: 0.1
    repetition_penalty: 1.1
prompt_format:
    system: "{instruction}\n"
    assistant: "### Response:\n{instruction}\n"
    trailing_assistant: "### Response:\n"
    user: "### Instruction:\n{instruction}\n"
    default_system_message: "Below is an instruction that describes a task. Write a response that appropriately completes the request."
    stopping_sequences: ["### Response:", "### End"]
```

aws NVIDIA

Cost

Quality

How all the pieces fit together



Anyscale Endpoints

LLMs served via API

LLMs fine-tuned via API

Anyscale Private Endpoints

Serve your LLMs from your Cloud

Fine-tune & customize in your Cloud

Anyscale AI Platform

AI app serving & routing

Model training & continuous tuning

Python-native Workspaces

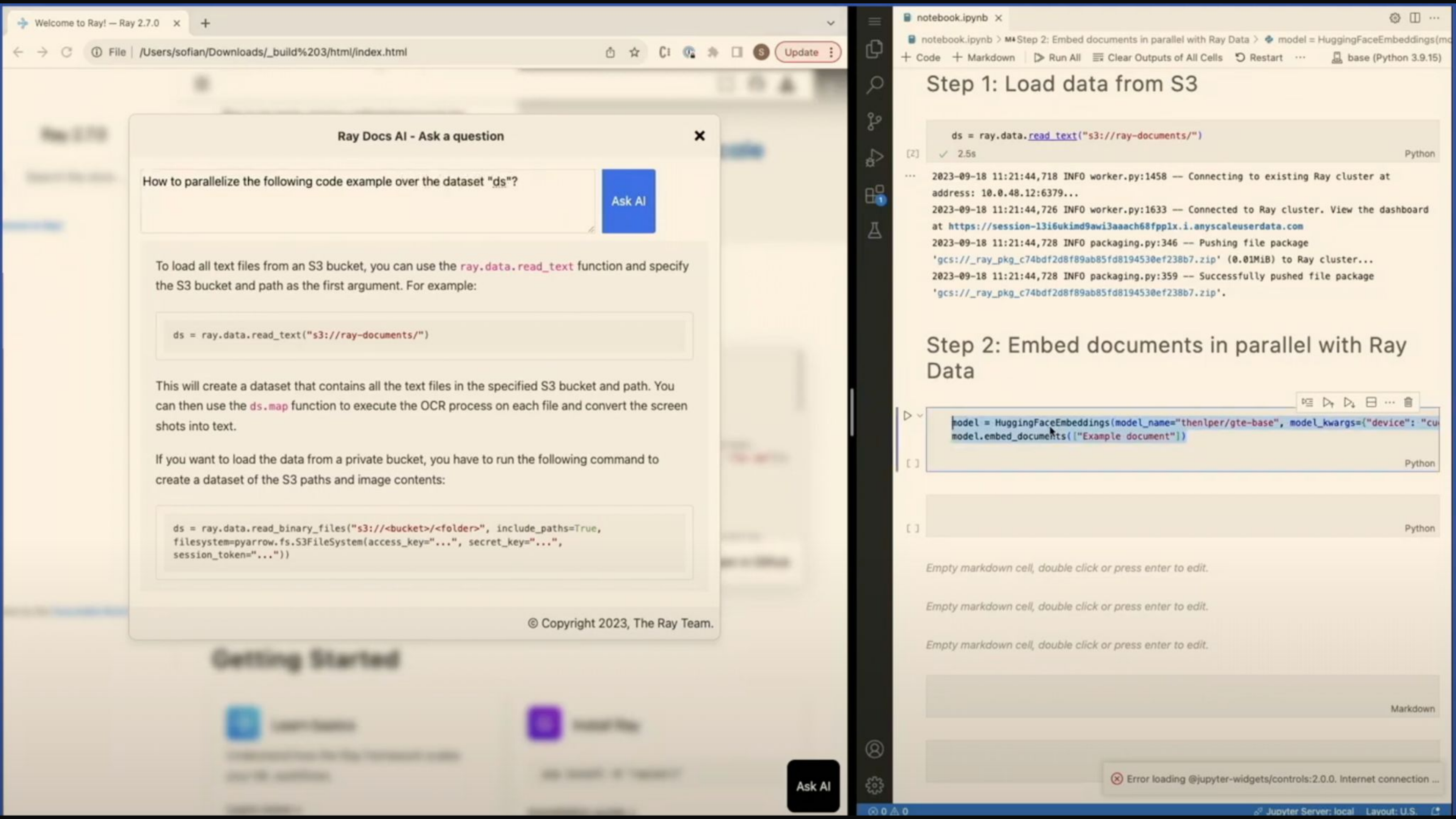
GPU/CPU optimizations

Multi-Cloud, auto-scaling

Ray Open Source

Ray AI Libraries

Ray Core


Welcome to Ray! — Ray 2.7.0
File
/Users/sofian/Downloads/_build%203/html/index.html
Update
notebook.ipynb
notebook.ipynb > Step 2: Embed documents in parallel with Ray Data > model = HuggingFaceEmbeddings(mo
+ Code
+ Markdown
Run All
Clear Outputs of All Cells
Restart
base (Python 3.9.15)
Ask AI
Jupyter Server: local
Layout: U.S.

Ray Docs AI - Ask a question

How to parallelize the following code example over the dataset "ds"?

Ask AI

To load all text files from an S3 bucket, you can use the `ray.data.read_text` function and specify the S3 bucket and path as the first argument. For example:

```
ds = ray.data.read_text("s3://ray-documents/")
```

This will create a dataset that contains all the text files in the specified S3 bucket and path. You can then use the `ds.map` function to execute the OCR process on each file and convert the screen shots into text.

If you want to load the data from a private bucket, you have to run the following command to create a dataset of the S3 paths and image contents:

```
ds = ray.data.read_binary_files("s3://<bucket>/<folder>", include_paths=True,
filesystem=pyarrow.fs.S3FileSystem(access_key="...", secret_key="...",
session_token="..."))
```

© Copyright 2023, The Ray Team.

Step 1: Load data from S3

```
ds = ray.data.read_text("s3://ray-documents/")
```

[2] ✓ 2.5s Python

```
2023-09-18 11:21:44,718 INFO worker.py:1458 -- Connecting to existing Ray cluster at
address: 10.0.48.12:6379...
2023-09-18 11:21:44,726 INFO worker.py:1633 -- Connected to Ray cluster. View the dashboard
at https://session-13i6ukimd9awi3aaach68fpp1x.i.anyscaleuserdata.com
2023-09-18 11:21:44,728 INFO packaging.py:346 -- Pushing file package
'gcs://_ray_pkg_c74bdf2d8f89ab85fd8194530ef238b7.zip' (0.01MiB) to Ray cluster...
2023-09-18 11:21:44,728 INFO packaging.py:359 -- Successfully pushed file package
'gcs://_ray_pkg_c74bdf2d8f89ab85fd8194530ef238b7.zip'.
```

Step 2: Embed documents in parallel with Ray Data

```
model = HuggingFaceEmbeddings(model_name="thenlper/gte-base", model_kwargs={"device": "cu
model.embed_documents(["Example document"])
```

[ ] Python

[ ] Python

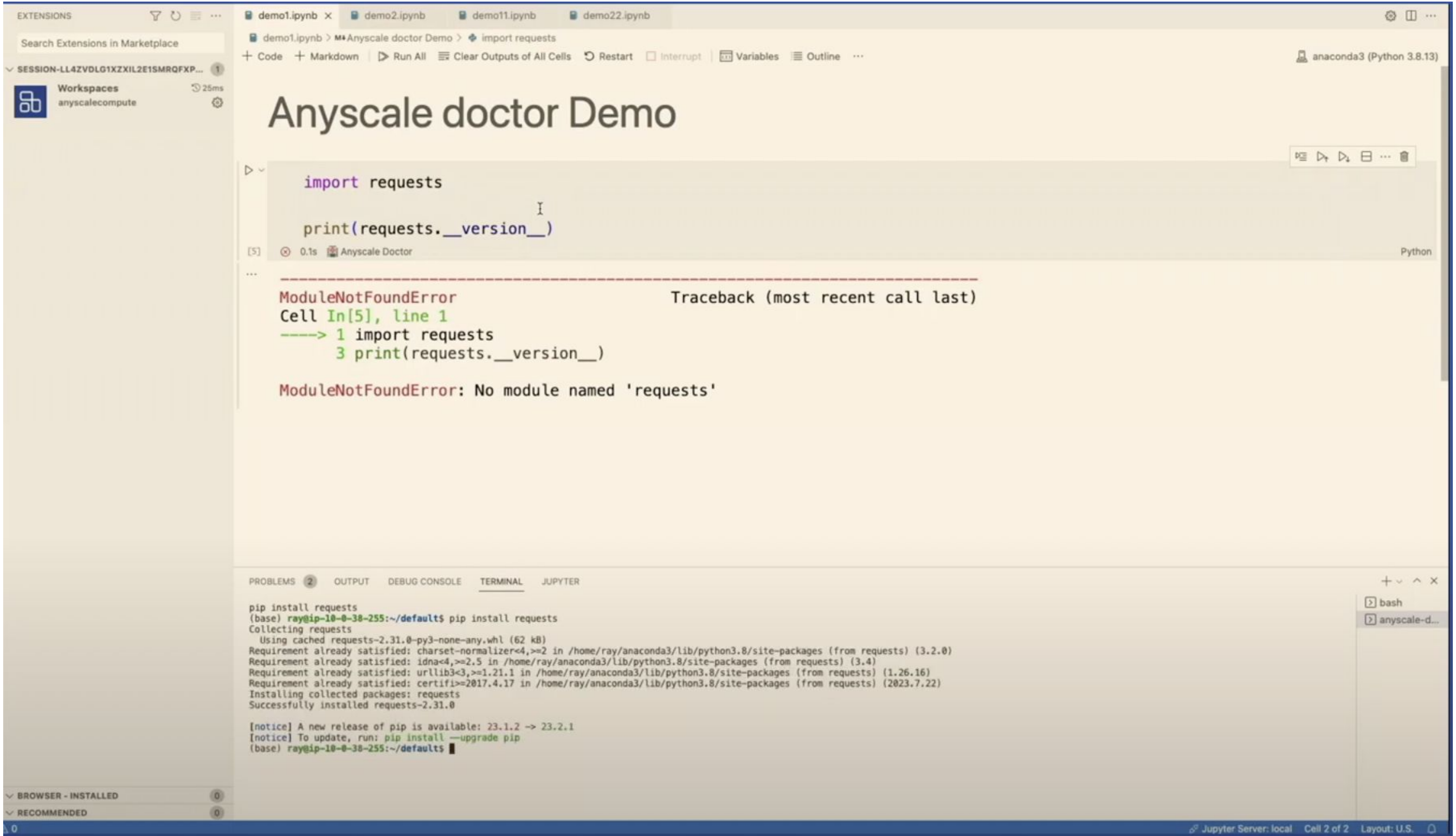
Empty markdown cell, double click or press enter to edit.

Empty markdown cell, double click or press enter to edit.

Empty markdown cell, double click or press enter to edit.

Markdown

Error loading @jupyter-widgets/controls:2.0.0. Internet connection ...

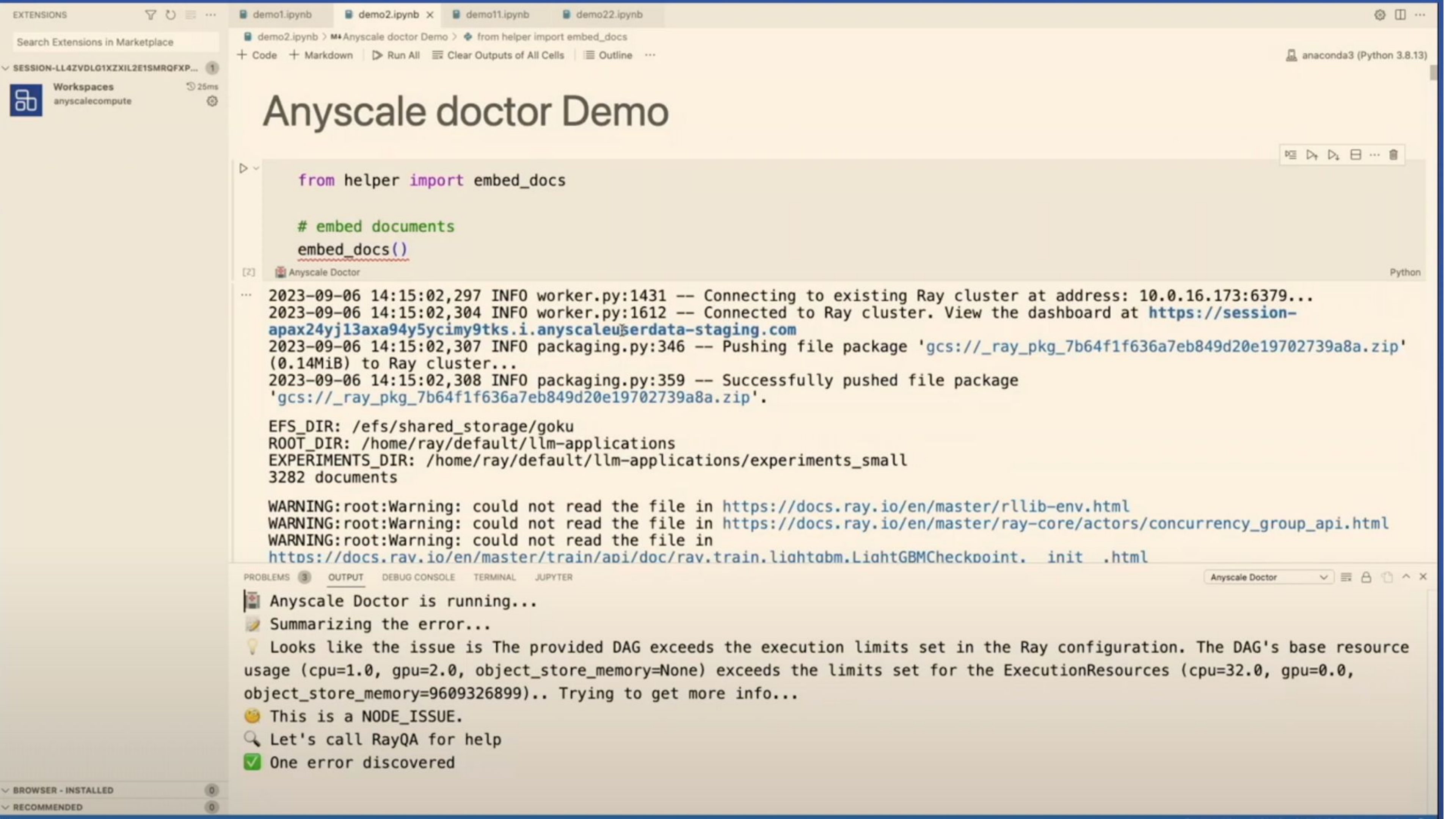


Anyscale doctor Demo

```python
import requests

print(requests.__version__)
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[5], line 1
----> 1 import requests
      3 print(requests.__version__)

ModuleNotFoundError: No module named 'requests'
```

```
pip install requests
(base) ray@ip-10-0-38-255:~/default$ pip install requests
Collecting requests
  Using cached requests-2.31.0-py3-none-any.whl (62 kB)
Requirement already satisfied: charset-normalizer<4,>=2 in /home/ray/anaconda3/lib/python3.8/site-packages (from requests) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in /home/ray/anaconda3/lib/python3.8/site-packages (from requests) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /home/ray/anaconda3/lib/python3.8/site-packages (from requests) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in /home/ray/anaconda3/lib/python3.8/site-packages (from requests) (2023.7.22)
Installing collected packages: requests
Successfully installed requests-2.31.0

[notice] A new release of pip is available: 23.1.2 -> 23.2.1
[notice] To update, run: pip install --upgrade pip
(base) ray@ip-10-0-38-255:~/default$ 
```



Anyscale doctor Demo

```python
from helper import embed_docs

# embed documents
embed_docs()
```

```
2023-09-06 14:15:02,297 INFO worker.py:1431 -- Connecting to existing Ray cluster at address: 10.0.16.173:6379...
2023-09-06 14:15:02,304 INFO worker.py:1612 -- Connected to Ray cluster. View the dashboard at https://session-
apax24yj13axa94y5ycimy9tks.i.anyscaleuserdata-staging.com
2023-09-06 14:15:02,307 INFO packaging.py:346 -- Pushing file package 'gcs://_ray_pkg_7b64f1f636a7eb849d20e19702739a8a.zip'
(0.14MiB) to Ray cluster...
2023-09-06 14:15:02,308 INFO packaging.py:359 -- Successfully pushed file package
'gcs://_ray_pkg_7b64f1f636a7eb849d20e19702739a8a.zip'.

EFS_DIR: /efs/shared_storage/goku
ROOT_DIR: /home/ray/default/llm-applications
EXPERIMENTS_DIR: /home/ray/default/llm-applications/experiments_small
3282 documents

WARNING:root:Warning: could not read the file in https://docs.ray.io/en/master/rllib-env.html
WARNING:root:Warning: could not read the file in https://docs.ray.io/en/master/ray-core/actors/concurrency_group_api.html
WARNING:root:Warning: could not read the file in
https://docs.ray.io/en/master/train/api/doc/ray.train.lightgbm.LightGBMCheckpoint.  init  .html
```

```
Anyscale Doctor is running...
Summarizing the error...
Looks like the issue is The provided DAG exceeds the execution limits set in the Ray configuration. The DAG's base resource
usage (cpu=1.0, gpu=2.0, object_store_memory=None) exceeds the limits set for the ExecutionResources (cpu=32.0, gpu=0.0,
object_store_memory=9609326899).. Trying to get more info...
This is a NODE_ISSUE.
Let's call RayQA for help
✅ One error discovered
```

Some quotes from our customers

Realchar.ai

"Realchar.ai is about delivering immersive, realistic experiences for our users, **not fighting infrastructure** or upgrading open source models. Endpoints made it possible for us to introduce **new services in hours**, instead of weeks, and for a **fraction of the cost of proprietary services**. It also enables us to seamlessly personalize user experiences at scale."

Merlin

"We use Anyscale Endpoints to power consumer-facing services that have reach to millions of users … Anyscale Endpoints gives us **5x-8x cost advantages over alternatives,** making it easy for us to make Merlin even more powerful while **staying affordable for millions of users**."

What makes it hard to go from demo to production?

Going from Demo to Production

- Not hallucinating
  (TL;DR use RAG)
- Knowing your relevance, consistency meets the bar
  (TL;DR automated evaluation using GPT-4)
- **Cost**
- **Data**
- **Vendor Lock-in**
- **Deploying LLMs**

Jerry
& Harrison
got you covered

From Demo to Production: Case Study

You provide software for summarizing long email threads

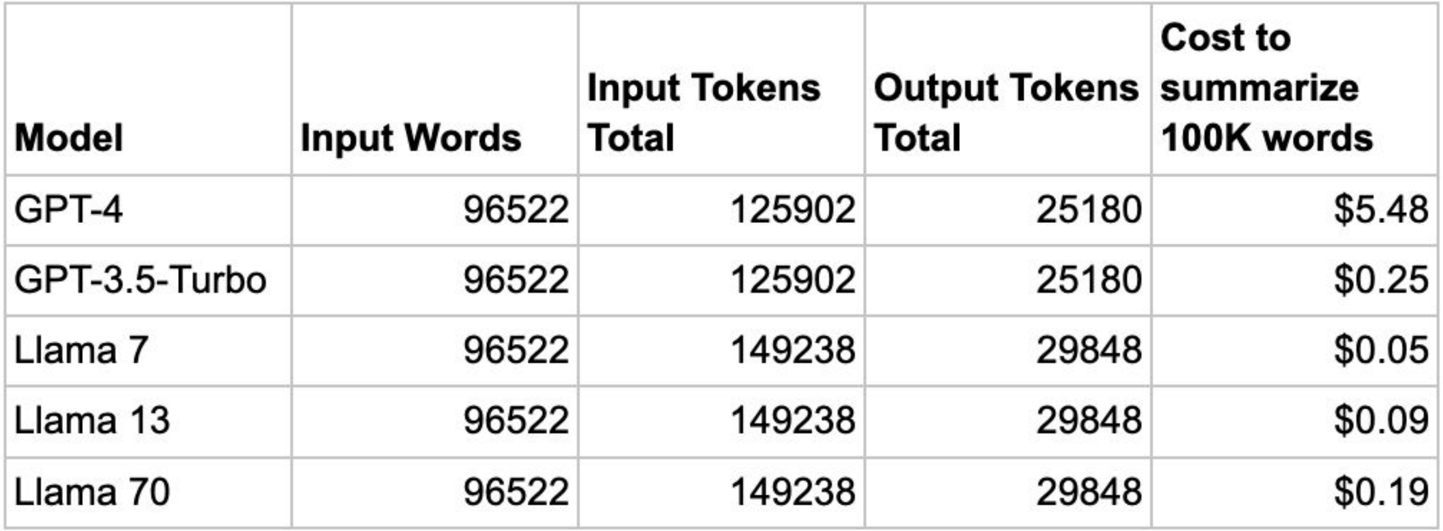
You’ve written this in LangChain and/or LlamaIndex

You use GPT-4 with “safe” emails – results look good!

What stops you from productionizing?

Cost

GPT-4 is **Expensive** – 30x Llama 70b for similar performance



Model	Input Words	Input Tokens Total	Output Tokens Total	Cost to summarize 100K words
GPT-4	96522	125902	25180	$5.48
GPT-3.5-Turbo	96522	125902	25180	$0.25
Llama 7	96522	149238	29848	$0.05
Llama 13	96522	149238	29848	$0.09
Llama 70	96522	149238	29848	$0.19

Cost

Onboarding (50,000 threads, each thread is 1,000 words)

How much to onboard each user to email summaries?

GPT-4: $2,740
Llama 2: $95

Daily (100 threads, each thread is 1,000 words)

GPT-4: $5.48
Llama 2: $0.19

Cost *and* Quality simultaneously?

LLMs good at summarizing (they trounce hand-built systems)

Most LLMs will produce text that coherent, fluent and relevant

What is the main differentiator between a good summary and a bad summary using LLMs?

Factual correctness

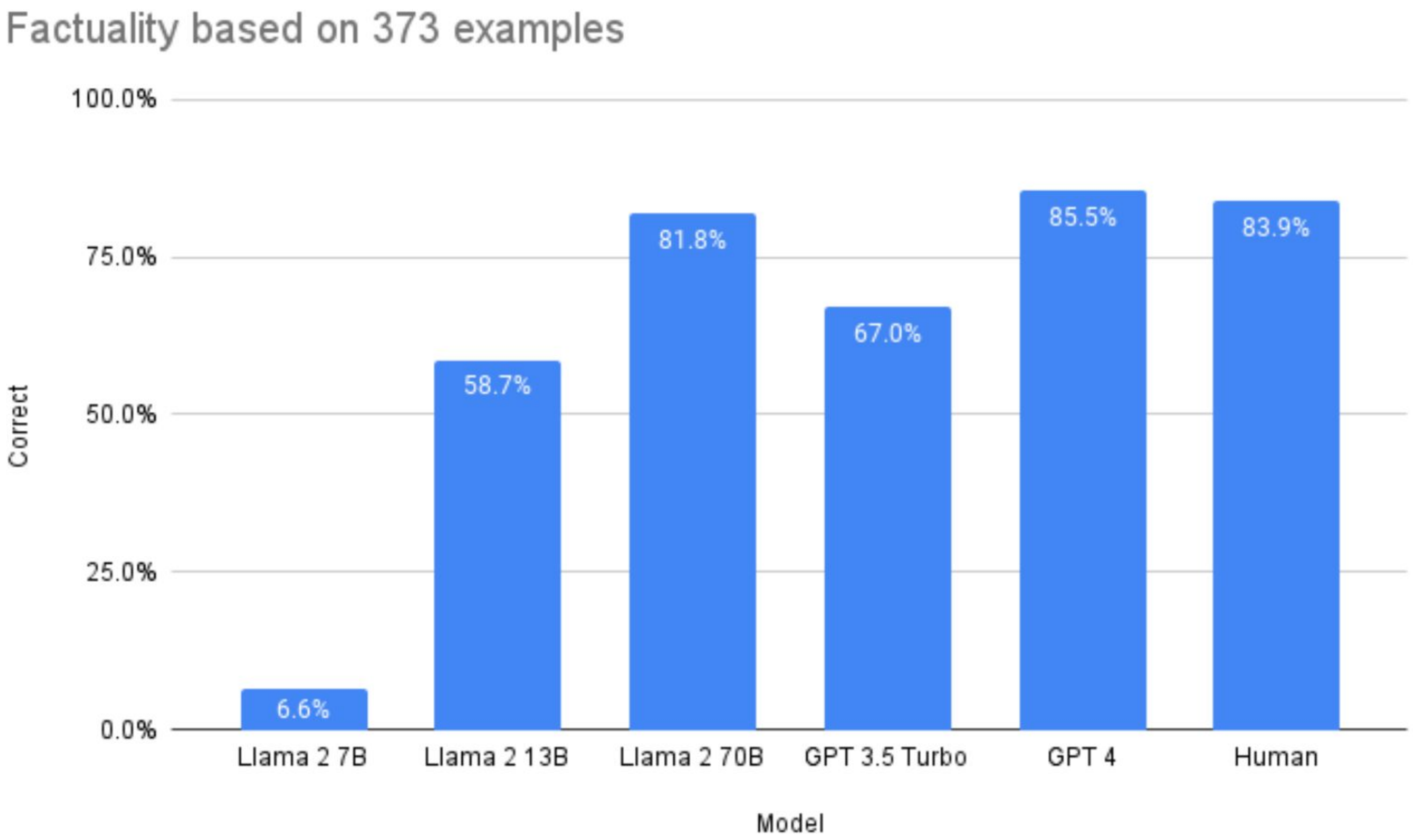
Example of comparable quality: Factuality eval

Summary Ranking established in literature.

“insiders say the row brought simmering tensions between the starkly contrasting pair -- both rivals for miliband's ear -- to a head.”

A: insiders say the row brought tensions between the contrasting pair.

B: insiders say the row brought simmering tensions between miliband's ear.


Factuality based on 373 examples
100.0%
75.0%
50.0%
25.0%
0.0%
Correct
6.6%
58.7%
81.8%
67.0%
85.5%
83.9%
Llama 2 7B
Llama 2 13B
Llama 2 70B
GPT 3.5 Turbo
GPT 4
Human
Model

Result

For the summarization task, LLama 70b is about as good as GPT-4 (on factuality)

Dropping to GPT-3.5-Turbo doesn’t work, significant drop in quality

Llama 2 70b costs 30x less

The Power of Fine-tuning in Cost Reduction

A small fine-tuned open source model

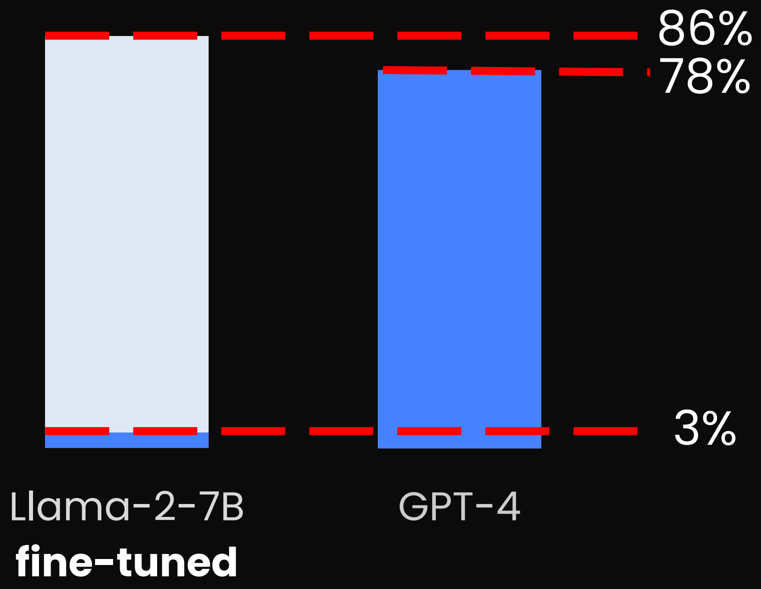
can outperform the best available general model

in some cases

Example Datapoint

Natural Language	Name the result for week less than 7 and game sites of los angeles memorial coliseum
CREATE TABLE	CREATE TABLE table_name_25 (result VARCHAR, week VARCHAR, game_site VARCHAR)
SQL Query	SELECT result FROM table_name_25 WHERE week < 7 AND game_site = "los angeles memorial coliseum"

Anyscale Endpoints - fine-tuning



Superior task-specific performance at **1/300th** the cost of GPT-4.

There is still a price to be paid

- APIs are easy
    - Anyscale Endpoints is OpenAI API compatible.
- But:
    - ChatGPT follows instructions
    - Llama 2 doesn’t always do as well
    - Hypothesis: OpenAI does a *lot* of RLHF (reinforcement learning w/ Human Feedback)
- We *still* use GPT-4 a lot
    - Hard queries
    - Evaluation

Instruction following

What we asked for: Please give an A or a B.

What we got from GPT-4: A

What we got from Llama 2 70b:

‘The correct answer is A: those who receive centrelink payments made up half of radio rental's income last year. Explanation: Summary A accurately summarizes the article sentence by mentioning that those who receive centrelink payments made up half of radio rental's income last year. It maintains the same meaning and information as the original sentence. On the other hand, Summary B is inconsistent with the article sentence. It suggests that the ABC's report only mentioned that those who receive centrelink payments made up radio rental's income last year, which is not entirely accurate. The article sentence explicitly states that the ABC reported that those who receive centrelink payments made up half of radio rental's income last year. Therefore, Summary A is the better choice’.

Function Templates

Convert the text below into one that calls a Python function.

The function is **find_flights(departure_city, arrival_city,time, date, class)**

Convert to the appropriate city code using another function **city_code(str)** that returns the city code for a given city.

“Hi. I'd like to book a flight to SF from Boston on Wednesday 20 September in the evening. Business class.”

Llama 13B output:

```
find_flights(Boston,
            San_Francisco,
            “2023-09-20”,
             “18:00”,
            “business”)
```

Does this parse?

- No, first two parameters are variables, should have quotes
- Didn’t use city_code function
- Decided 6pm was evening

Vs OpenAI strictly defined templates

```
"functions": [{
  "name": "find_flights",
  "description": "template to find flights.",
  "parameters": {
     "type": "object",
     "properties": {
        "from_city_code": {
           "type": "string",
           "description": "Three letter code for the city"
      }, ...
```

vs Proprietary (OpenAI)

```
find_flights(city_code(“Boston”),
             city_code(“San Francisco”),
              “2023-09-20”,
             “evening”,
             “business”)
```

Data and Privacy

Are you comfortable sending your company’s emails to OpenAI? Not trying to FUD but customers have expressed concerns.

Are you worried about explaining to your users that you are sending the data to OpenAI?

Are there restrictions you have (e.g. in-country requirements or GDPR) that apply?

Making some tough decisions

Proprietary Pros

- Best quality models overall
- Simplicity
- Better instruction following
- Newest features
  - Large Context Windows
  - Function Templates

Open Model pros

- Many more options
- Cost saving via:
  - Cheaper models
  - Fine tuning benefits
- Deployment flexibility help w/ data + privacy
- No vendor lock-in

Deployment complexity

Consequence of the first two problems

Where and how do you run your LLMs if not OpenAI

LLMs are not always small models

Llama 2 70b takes 4x A100 80GB GPUs to deploy

Self-Hosted Software

github.com/ray-project/ray-llm

Open source

Built on top of Ray Serve

Supports:

- Streaming
- Autoscaling

vLLM also good, but only single machine

text-generation-inference used to be open source, but since 0.94 closed

Self hosted Llama 2 Models

Llama 2 7B: One g5.2xlarge is ~$7000/yr

- Can do ~700 tokens/s
- No autoscaling or redundancy

Llama 2 70B:

- You need 4x A100 80GB – if you can get them
- Lambda Labs: $2/GPU so we're talking $70,000/yr.
- Break-even point: 70 billion tokens vs public endpoints

“Assisted” Self-Hosted: A new option

Anyscale Private Endpoints

Give us AWS credentials (locked down)

We run in your cloud

Handles autoscaling, observability, updates etc

Public Open Source Serving

Anyscale Endpoint ($1 / million tokens)
Fireworks.ai (~about the same price)

Offer OpenAI compatible APIs

More flexibility, e.g. can serve fine tuned models at different sizes

Concluding on the difficulties

- Cost is a major concern.
- Open Models give you options for saving money without decreasing quality, though they are not as polished.
- Open Models give you control on data – the AI can come to your data instead of the data going to your AI.
- Range of LLM deployment options each with different pros and cons (self-hosted, assisted self-hosted, managed, public)

What does the future hold?

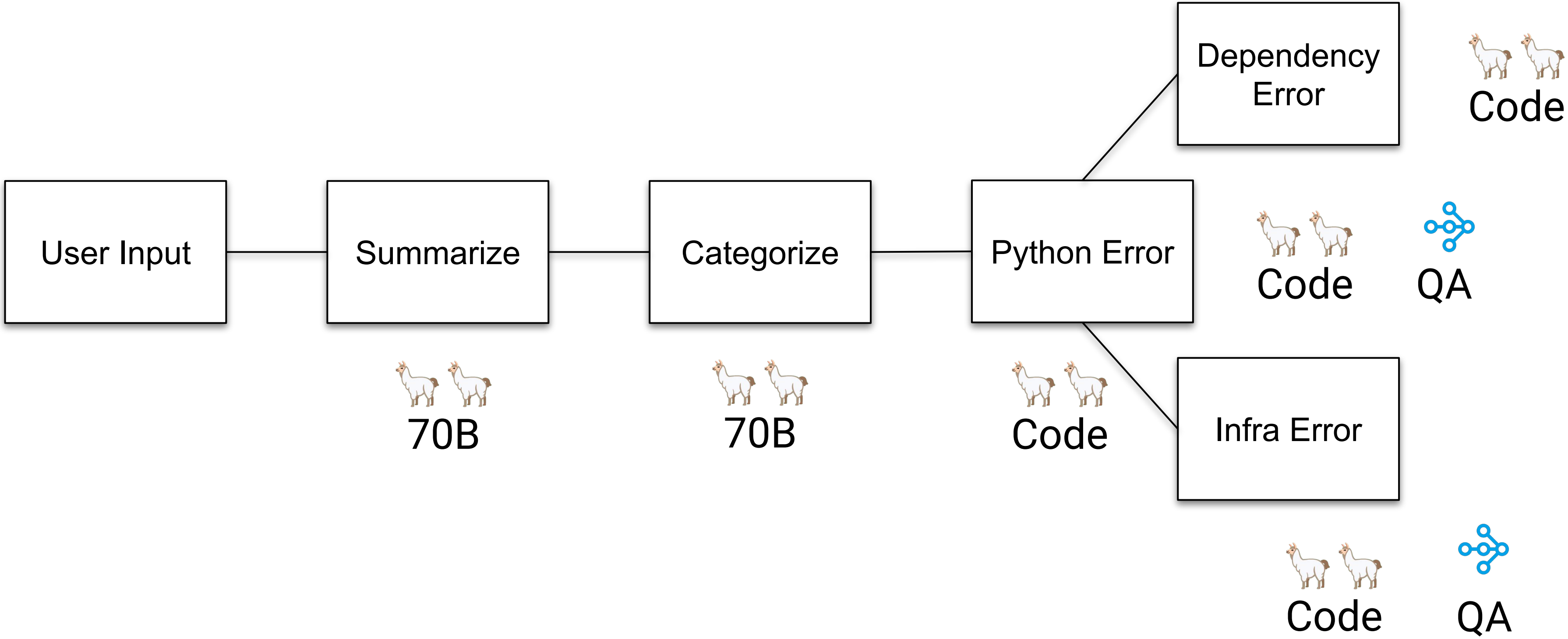
3 things I predict are coming

One task requires many LLM calls and many LLMs
(fine-tuned vs general, small vs large, open vs proprietary)

RAG will be default use of LLMs in enterprises

We haven't fully solved the model improvement over time.
Someone has to crack this

Anyscale Doctor



User Input

Summarize

🦙🦙
70B

Categorize

🦙🦙
70B

Python Error

🦙🦙
Code

Dependency Error

🦙🦙
Code

🦙🦙
Code

QA

Infra Error

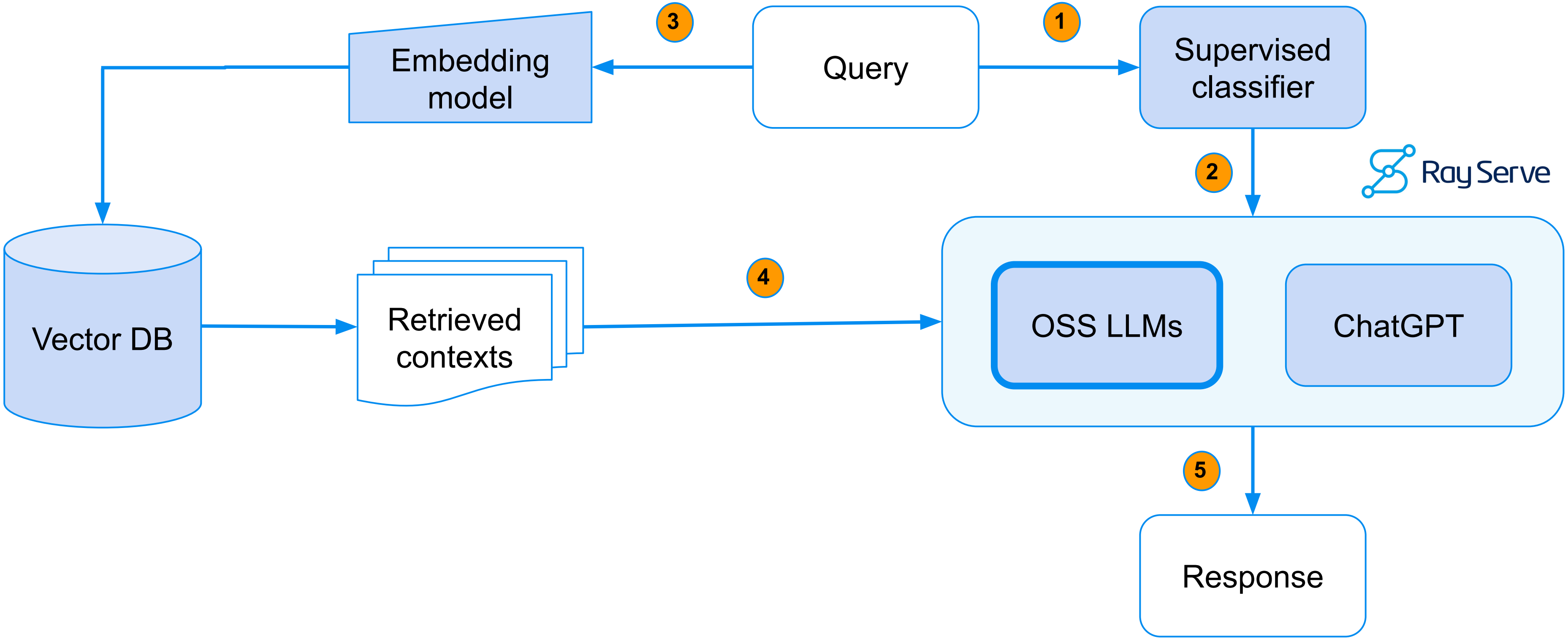
🦙🦙
Code

QA

Retrieval Augmented Generation

- Separate *knowledge* from *synthesis*
- Use search, vector database, tools etc for factual information
- Fine tuning does *not* help with facts
- Still some rough points with RAG
  - Precision and Recall of search results
  - Expensive because of long context

Ray Assistant



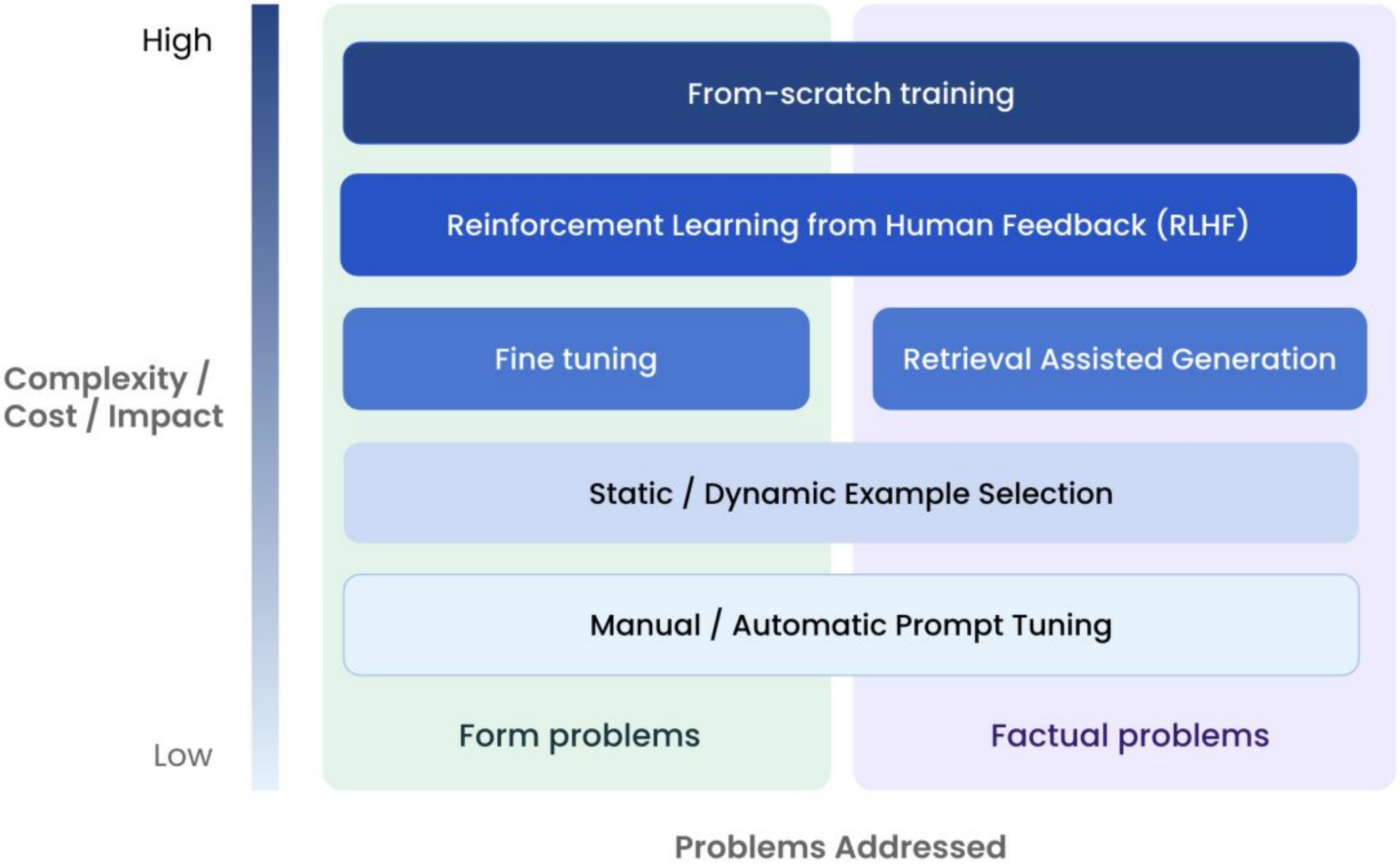
Key Takeaways

- LLMs: easy to demo, hard to productionize
- Challenges
  - Cost
  - Data, Privacy
  - Deployment complexity
  - Future optionality
- What does the future hold?
  - Applications are not a single LLM call
  - RAG will be the default way enterprises use LLMs
  - Not enough thought so far into closing the loop

Not enough thought so far into closing the loop

- How do you *correct* an LLM?
- Definitely not fine tuning
- RAG … but that feels indirect
- A corrections vector DB?
- Need to really think more broadly around domain specific model refinement

Space of Domain Specific Model Refinement (DSMR) techniques



High

Complexity / Cost / Impact

Low

From-scratch training

Reinforcement Learning from Human Feedback (RLHF)

Fine tuning

Retrieval Assisted Generation

Static / Dynamic Example Selection

Manual / Automatic Prompt Tuning

Form problems

Factual problems

Problems Addressed

Bonus: Waleed’s Hard-won Heuristics

1. Prototype with GPT-4 (or Claude if you need big context windows). If GPT-4 doesn’t work, nothing else is likely to.
2. One LLM call does one job. Don’t ask an LLM to summarize and classify. Do 2 llm calls, one to summarize one to classify.
3. Llama 2 70b can be useful as a “day to day” LLM if you remember Rule 2. GPT-4 is less sensitive to dual tasks.
4. Fine tuning is for form, not facts. RAG is for facts.
5. If you can, avoid self-hosting. It’s more difficult than it looks, esp multi-GPU LLMs like Llama 70b. If you have to, use ray-llm.

Key Takeaways

- LLMs: easy to demo, hard to productionize
- Learned a few lessons we want to share
- Challenges & overcoming them
  - Cost
  - Data, Privacy
  - Deployment complexity
- What does the future hold so we can prepare?
  - Applications are not a single LLM call
  - RAG will be the default way enterprises use LLMs
  - Fine tuning for cost reduction

Thank You!

Endpoints: endpoints.anyscale.com
RayLLM: github.com/ray-project/ray-llm
Details: anyscale.com/blog
Numbers: llm-numbers.ray.io
Ray: ray.io
Anyscale: anyscale.com

Me: mwk@anyscale.com