RAY SUMMIT 23

LLM

Practical Data and Evaluation Considerations for Building Production-Ready LLM Applications with LlamaIndex and Ray

Simon Suo
LlamaIndex

Goku Mohandas
Anyscale

Amog Kamsetty
Anyscale

presented by anyscale

🧑‍💻 Meet the TAs!

Artur

Philipp

The Plan

Here’s what to expect today.

Today’s agenda.

- Prototype
- Production Challenges & Diagnosis
- Evaluate
- Experiment & Optimize
- Deploy & Scale

✅ Tech check.

Participating via app.sli.do

- Join with code **#ray-llama**
- Ask questions.
  - Pose your own and upvote others.
  - TAs will be answering questions on a rolling basis.

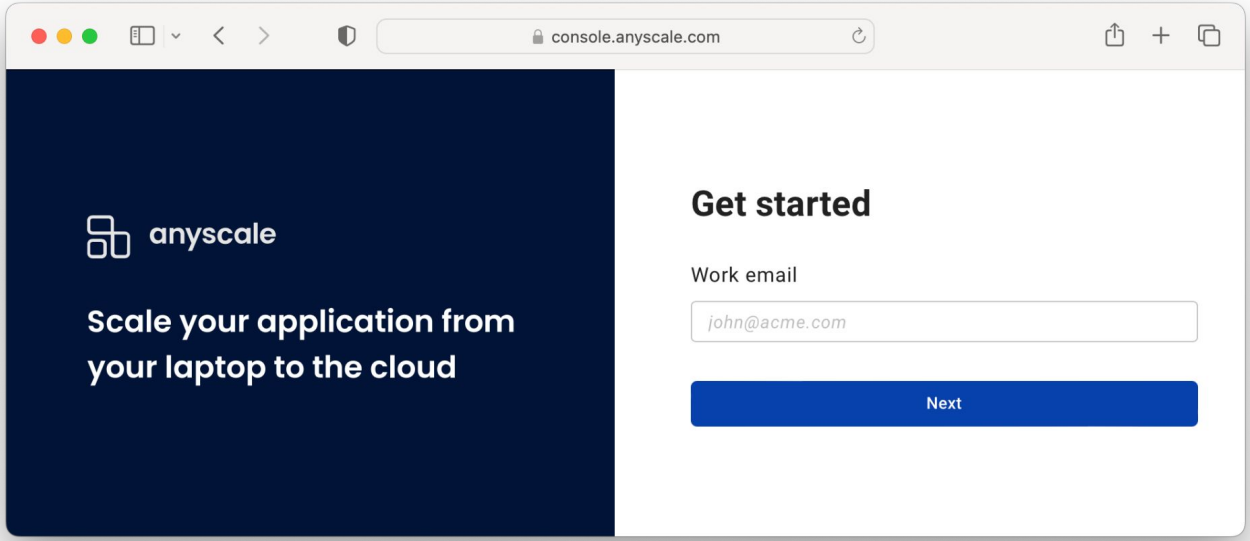
✅ Tech check.

 Accessing Anyscale clusters.

- All work will be in Anyscale provisioned clusters.
- Our GitHub repo will be mounted automatically.
- Access begins now.
  - Check your email for login information.
  - Step-by-step instructions to follow.

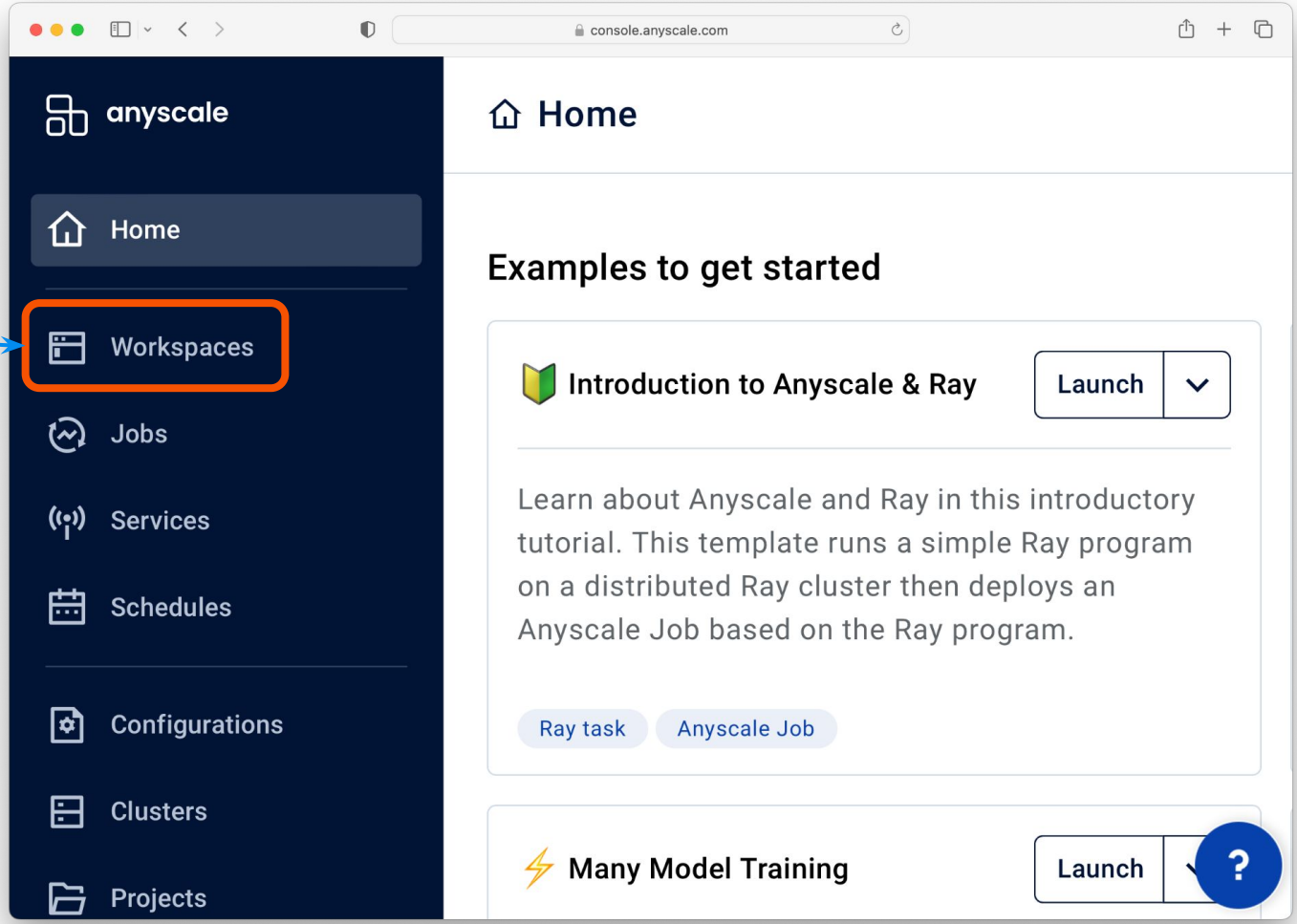
Anyscale login

Link to Anyscale cluster: console.anyscale.com

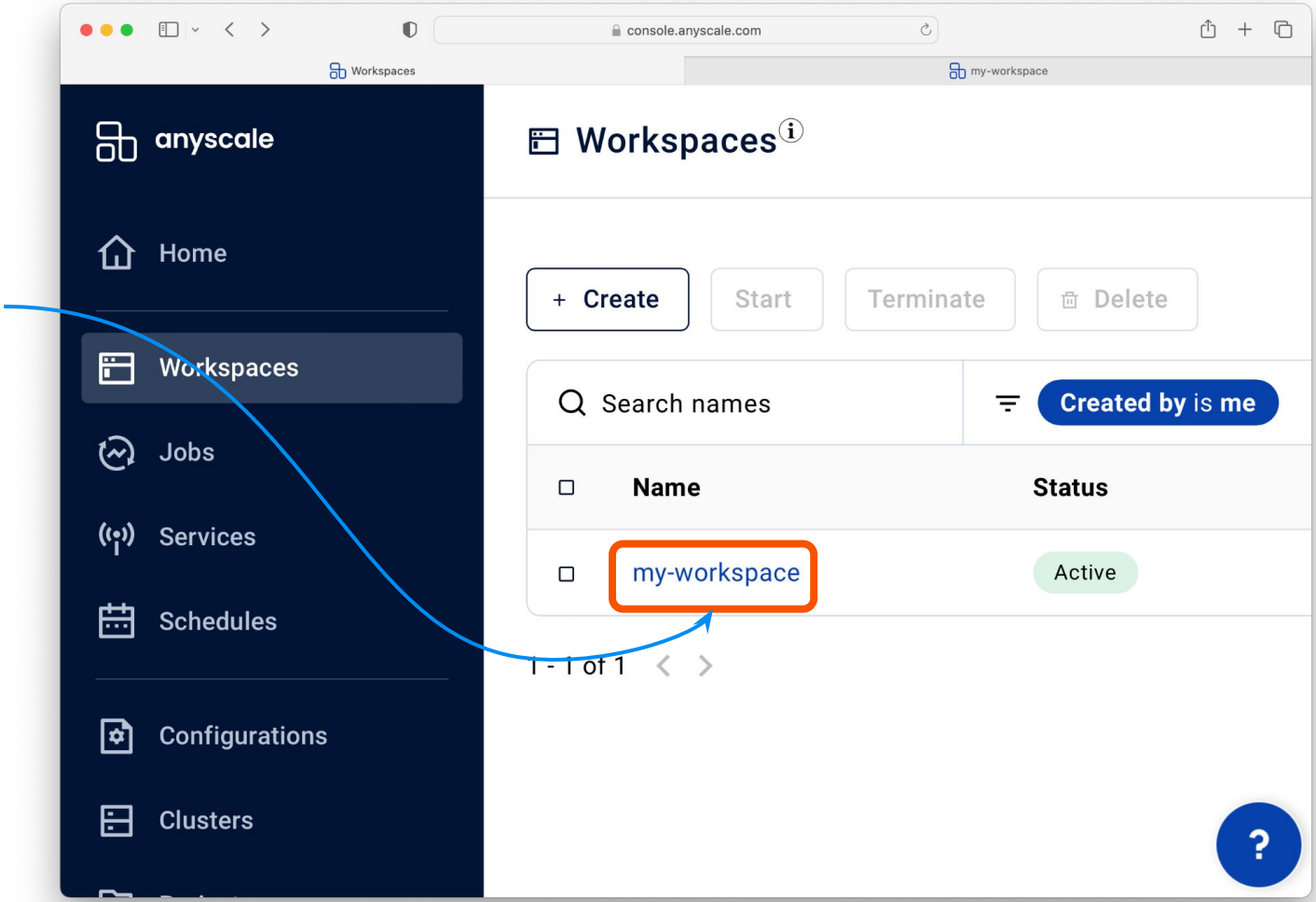


Enter the **unique credentials** sent to your email!

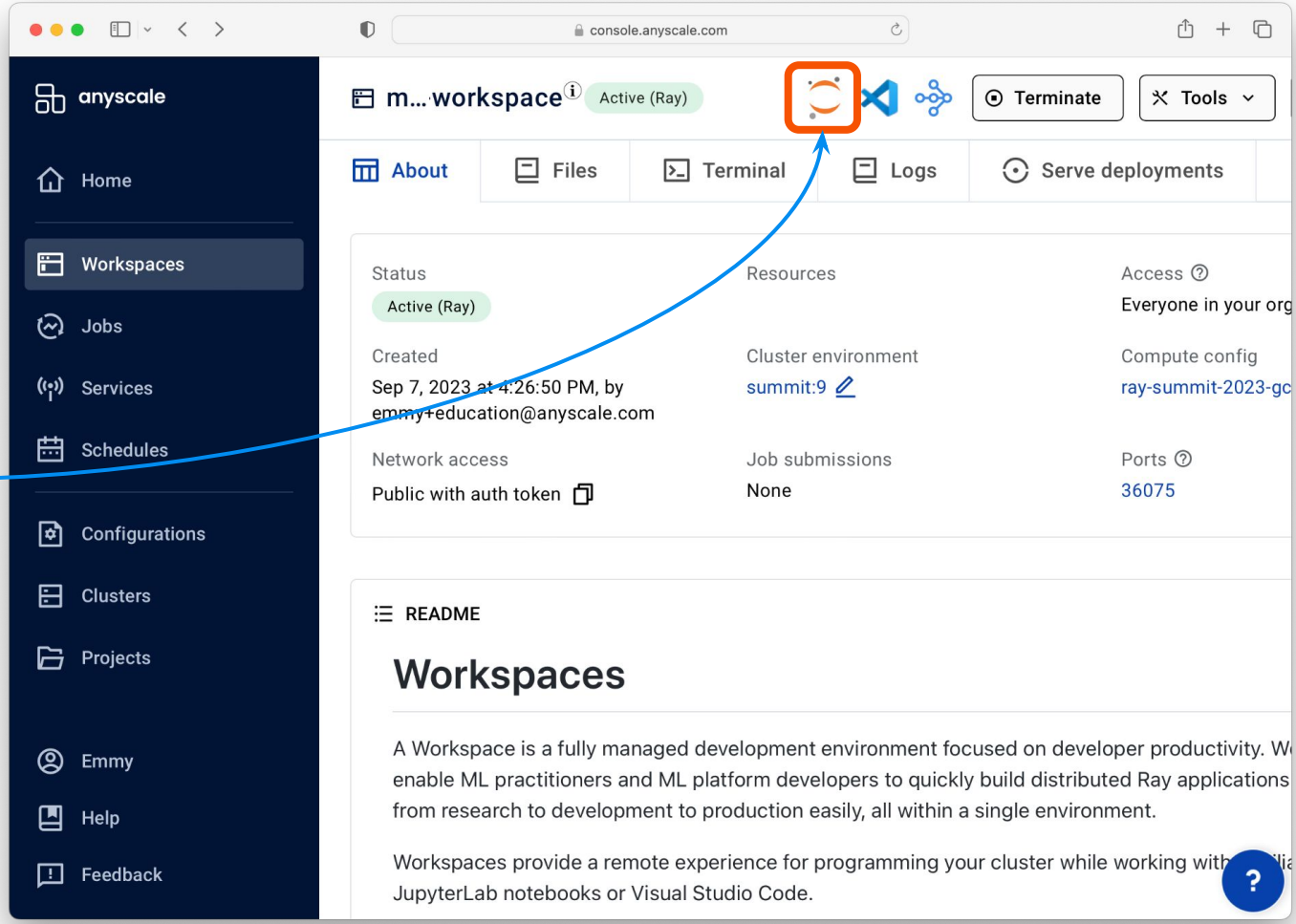
1. Select Workspaces



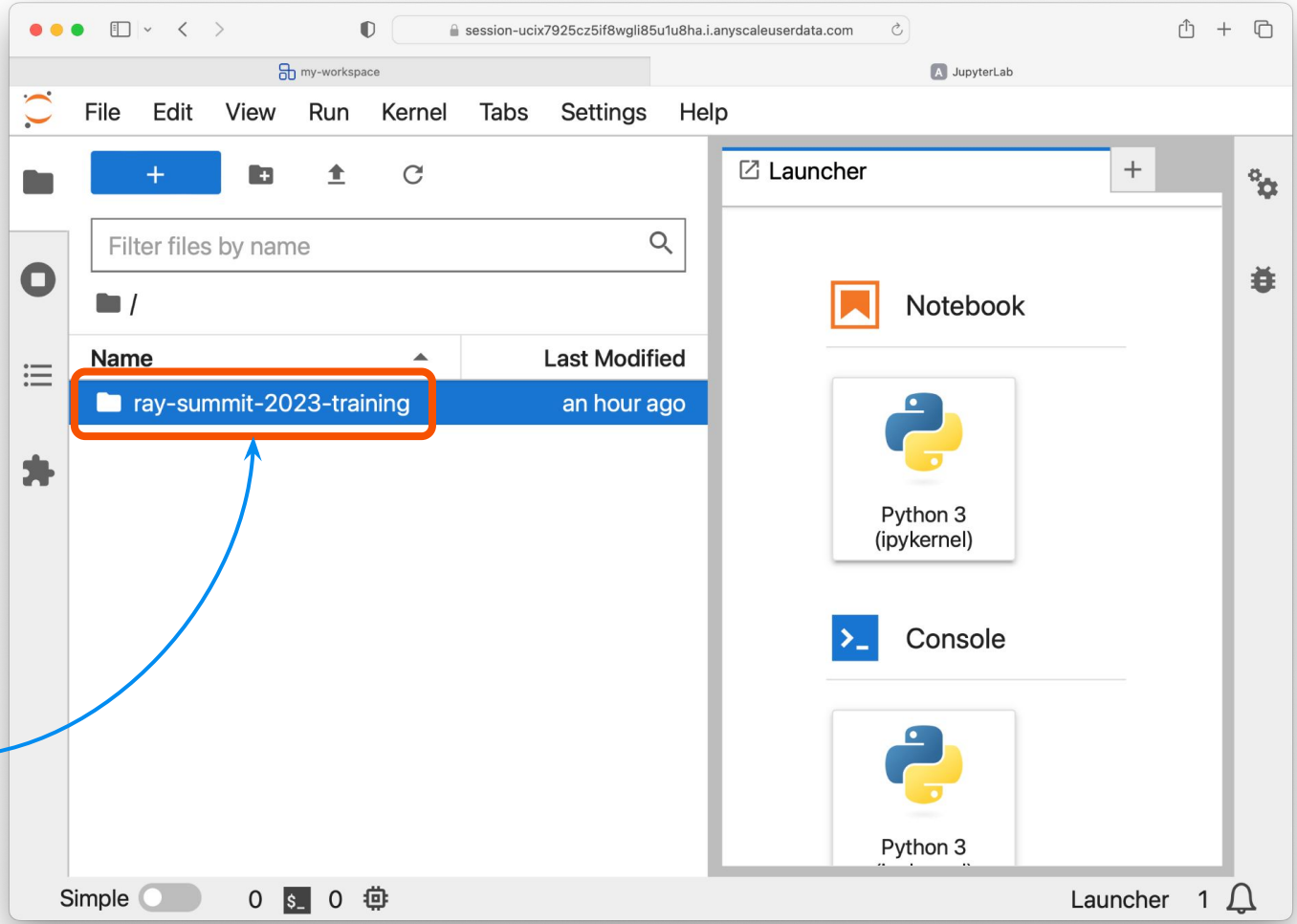
2. Select Your Workspace



3. Click on Jupyter icon



4. Find the content for your class here.



Prototype

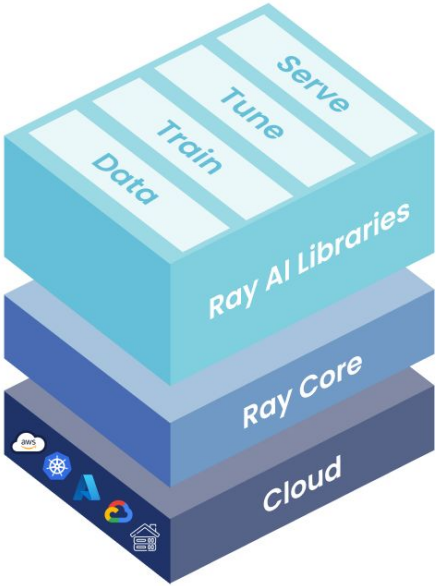
RAG System with LlamaIndex & Ray

LlamaIndex

Data Framework for LLM Applications

- Ingestion, indexing, and querying
- Orchestrates over LLMs, embedding models, vector DBs, graph DBs
- RAG, chat, data agents

Ray



high-level libraries that enable simple scaling of AI workloads

a low-level distributed computing framework with a concise core and Python-first API

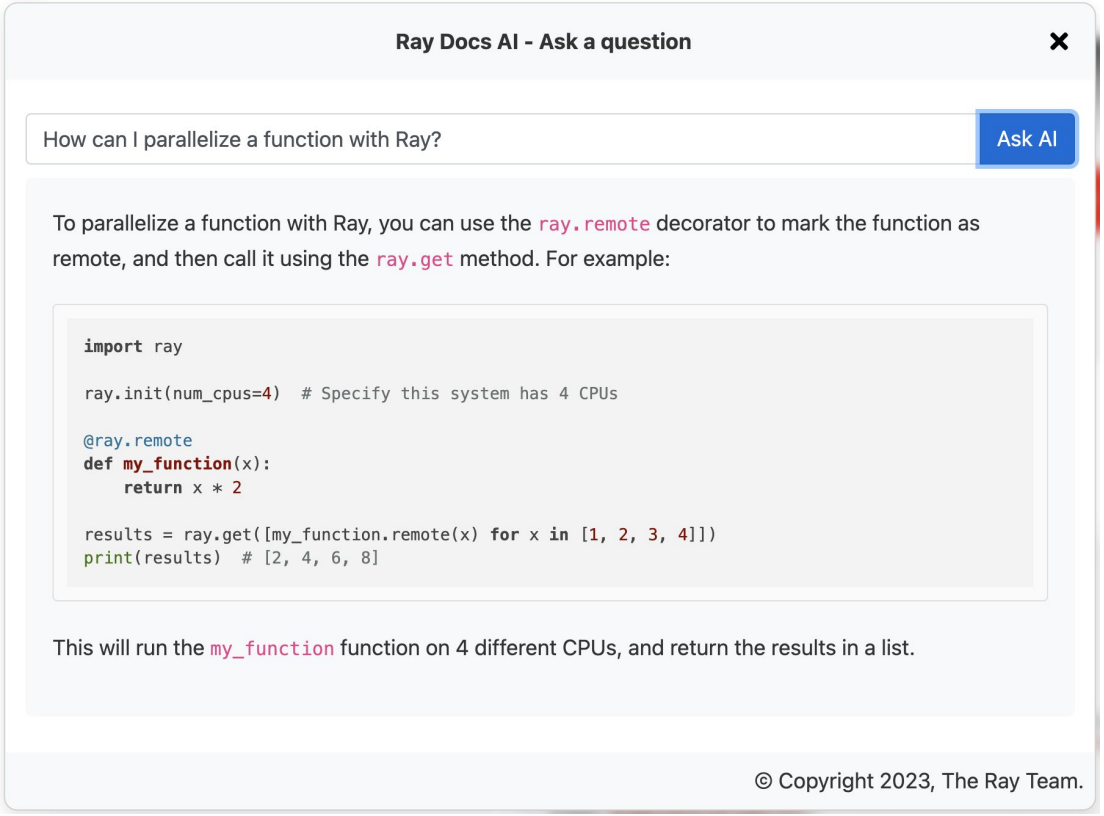
Scale the entire ML pipeline

- Scalable embedding generation & indexing with **Ray Data**
- Distributed training & fine-tuning with **Ray Train**
- Scalable application deployments with **Ray Serve**

RAY

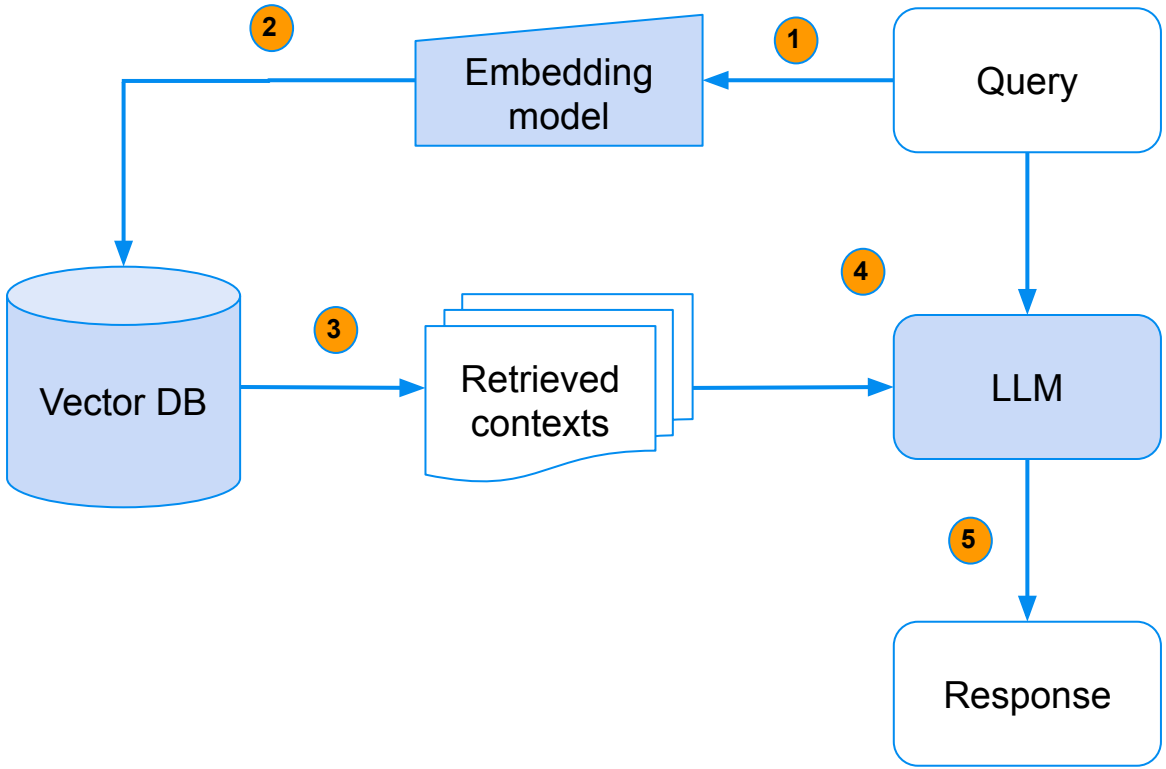
What are we building?

Ray Assistant

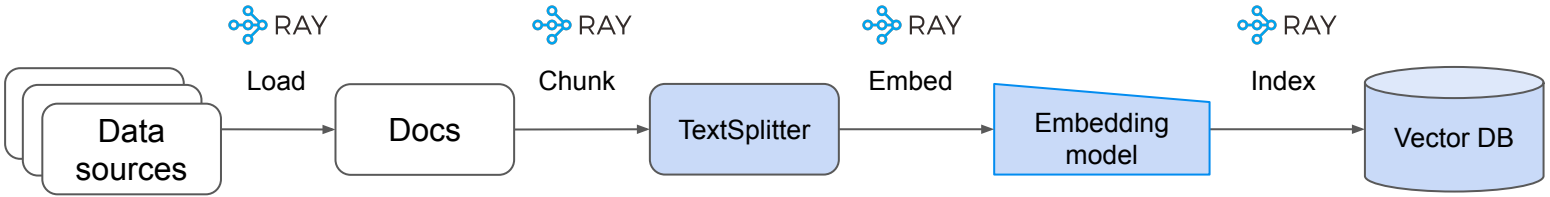


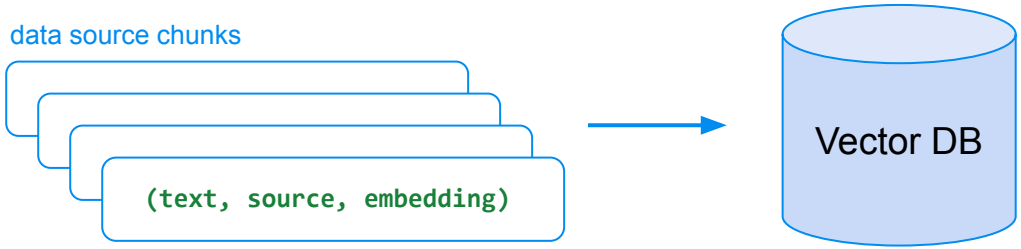
Base LLMs

RAG


2
Embedding model
1
Query
Vector DB
3
Retrieved contexts
4
LLM
5
Response

Creating our Vector DB





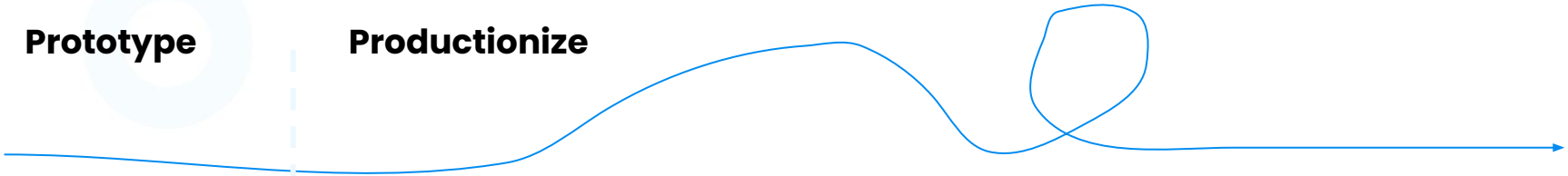
Lab

Let’s dive right in!

Production Challenges

Diagnosis & mitigation

From Prototype to Production



Few lines of code

latency

cost

hallucination

poor retrieval

output parsing errors

harmful answers

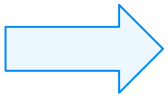
Need for evaluation, diagnosis, and optimization!

RAY

Challenges - non quality related

Symptoms

- Latency, rate-limits
- Cost
- Service Availability



Diagnosis & Mitigation

- Logging & monitoring
- Isolate issue to retrieval vs. generation components
- Evaluate different LLM service providers
- Use smaller, task-specific models
- Host your own models

Challenges - quality related

Symptoms

- Unknown performance
- Hallucinations
- Incomplete answers
- Poor retrievals

Diagnosis & Mitigation

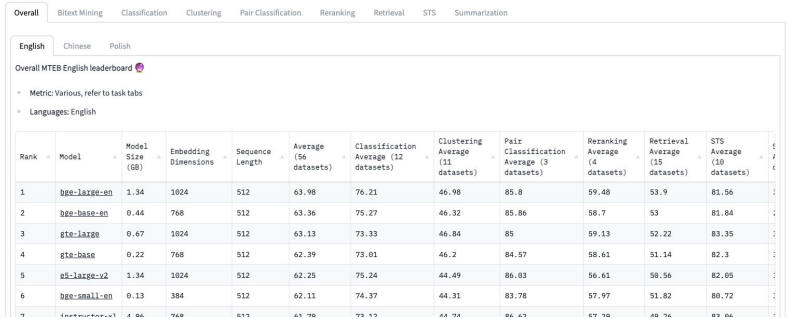
- **Evaluation**
  - Collect labels
  - Collect user feedback
- **Optimization**
  - Tune configs
  - Customization & fine-tune models

Evaluate

Understanding system performance

Expectation vs. Reality of Evaluation

- “Academic benchmark is all you need”
- Build, evaluate, done
- A single metric number that perfectly tell you the performance



Overall | Bitext Mining | Classification | Clustering | Pair Classification | Reranking | Retrieval | STS | Summarization

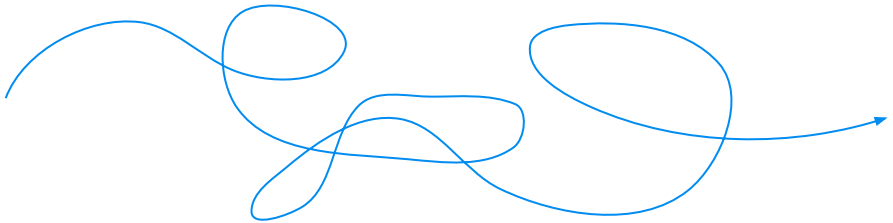
English | Chinese | Polish

Overall MTEB English leaderboard

- Metric: Various, refer to task tabs
- Languages: English

Rank	Model	Model Size (GB)	Embedding Dimensions	Sequence Length	Average (56 datasets)	Classification Average (12 datasets)	Clustering Average (11 datasets)	Pair Classification Average (3 datasets)	Reranking Average (4 datasets)	Retrieval Average (15 datasets)	STS Average (10 datasets)
1	bge-large-en	1.34	1024	512	63.98	76.21	46.98	85.8	59.48	53.9	81.56
2	bge-base-en	0.44	768	512	63.36	75.27	46.32	85.86	58.7	53	81.84
3	gte-large	0.67	1024	512	63.13	73.33	46.84	85	59.13	52.22	83.35
4	gte-base	0.22	768	512	62.39	73.01	46.2	84.57	58.61	51.14	82.3
5	e5-large-v2	1.34	1024	512	62.25	75.24	44.49	86.03	56.61	50.56	82.05
6	bge-small-en	0.13	384	512	62.11	74.37	44.31	83.78	57.97	51.82	80.72

- I don’t have user data yet?
- Should I label some data?
- “Vibe-check” only
- I can’t run big evaluation on every CI run
- When should I evaluate?



RAY

The development process

Idea Validation

Build prototype with toy data

Solution Validation

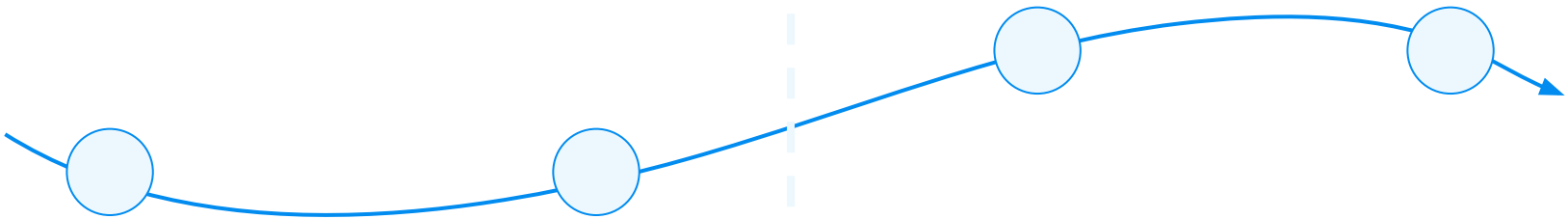
Build prototype with full corpus

Deployment

Deploy system & collect user queries & feedback

Optimization

Collect labels, improve system



RAY

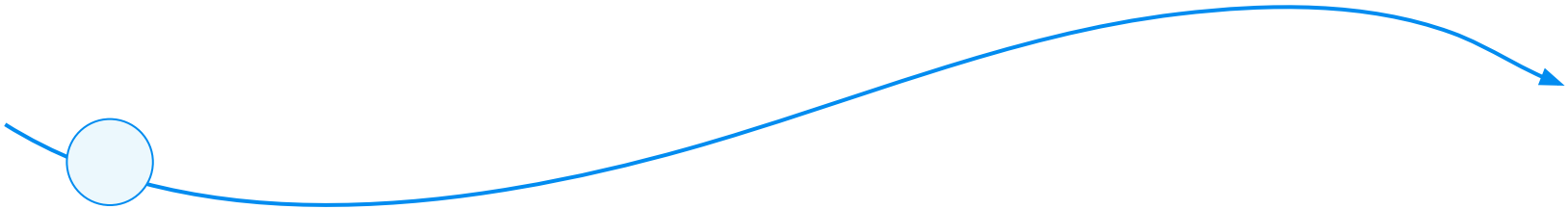
Evaluation in the development process

Idea Validation

Build prototype with toy data

At this stage

- Quick iteration cycle is critical
- “Vibe check” on ad-hoc queries
- Experiment with different modules
- Ad-hoc configuration changes



The “Vibe Check”

Ad-hoc spot check with random questions.

- Quick way to sanity check and iterate
- Good for initial prototyping and early development
- Could be surprising good indication of performance

Not systematic, but useful!

> Before we place a model in front of actual users, we like to test it ourselves and get a sense of the model's "vibes". The HumanEval test results we calculated earlier are useful, but there's nothing like working with a model to get a feel for it, including its latency, consistency of suggestions, and general helpfulness. Placing the model in front of Replit staff is as easy as flipping a switch. Once we're comfortable with it, we flip another switch and roll it out to the rest of our users.

https://blog.replit.com/llm-training

Finally, sometimes the best eval is human eval aka vibe check. (Not to be confused with the poorly named code evaluation benchmark HumanEval.) As mentioned in the Latent Space podcast with MosaicML (34th minute):

> The vibe-based eval cannot be underrated. ... One of our evals was just having a bunch of prompts and watching the answers as the models trained and see if they change. Honestly, I don't really believe that any of these eval metrics capture what we care about. One of our prompts was “suggest games for a 3-year-old and a 7-year-old to play” and that was a lot more valuable to see how the answer changed during the course of training. — Jonathan Frankle

https://eugeneyan.com/writing/llm-patterns/

The development process

Idea Validation

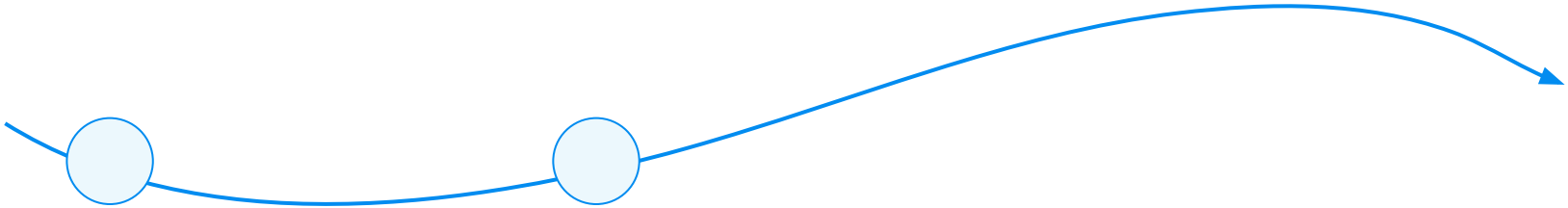
Build prototype with toy data

Solution Validation

Build prototype with full corpus

At this stage

- “Vibe check” on curated/representative set of test queries
- Towards more **systematic evaluation** to gain confidence towards initial deployment

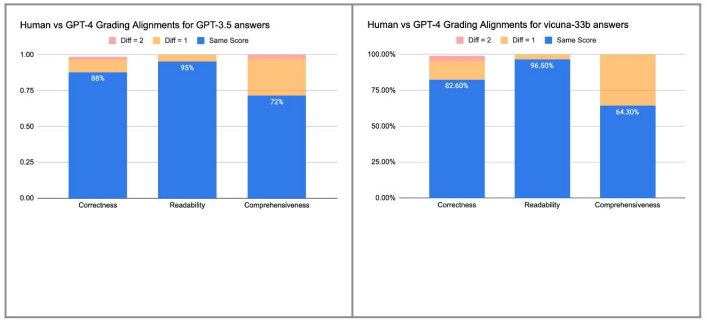


Challenge of Systemic Evaluation

- Metrics
  - Flexibility of natural language, no one right answer
  - Human evaluation is not scalable
- Data availability
  - Labeled data is slow & costly to collect
- Actionable insight
  - Not just “it’s bad”, but also “how to improve”
  - End-to-end vs. Component-wise

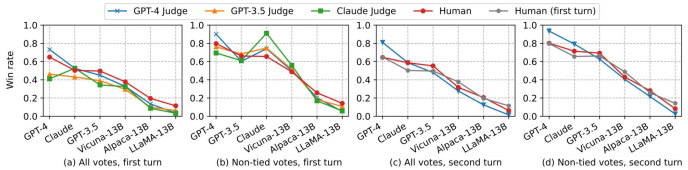
LLM-as-a-Judge

- Strong LLMs (GPT-4, Claude 2) are good evaluators
  - High agreement with human
  - scalability
  - Interpretability



https://www.databricks.com/blog/LLM-auto-eval-best-practices-RAG

- Approaches
  - Pairwise comparison
  - Single answer grading
  - Reference-guided grading (i.e. “golden” answer)



https://arxiv.org/pdf/2306.05685.pdf

RAY

Systematic Evaluation - Overview

End-to-end evaluation

Helps understand how well the **full RAG application** works:

- given user question, how "good" is the answer

Analogous to **integration** tests

Component-wise evaluation

Helps attribute quality issue to specific components:

- Retrieval: are we retrieving the relevant context
- Generation: given context, are we generating an accurate and coherent answer

Analogous to **unit** tests

Data for Systematic Evaluation

User Query

- representative set of real user queries

User Feedback

- Feedback from past interaction
- up/down vote, rating
- On retrieval and/or generation

“Golden” Context

- Set of relevant documents from our corpus to best answer a given query

“Golden” Answer

- Best answer given “golden” context
- Can be optional

RAY

Data Challenges

User Query	User Feedback	“Golden” Context	“Golden” Answer
• Need to deploy system & collect	• Need to deploy system & collect	• Need labelers	• Need labelers
Relatively easy	**Require good UX for good data**	**Relatively cheap/easy**	**More costly/tedious**

“Cold Start” Problem

Dilemma

We have the chicken and egg problem of:

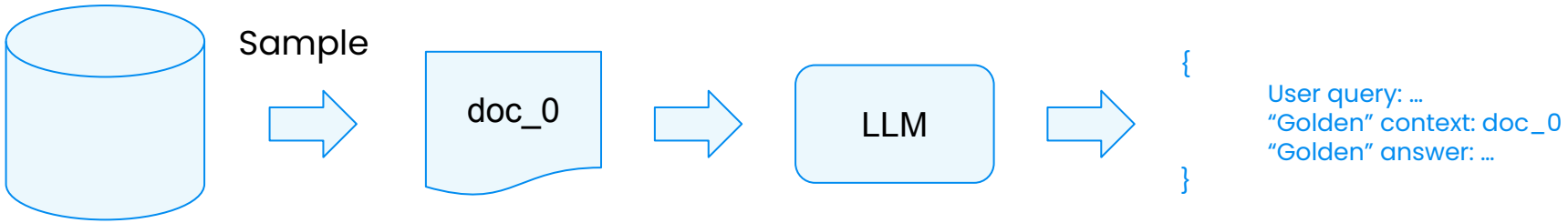
1. We want to evaluate to gain confidence of our RAG system before deployment
2. Without deployment, we can’t collect real user queries and label an evaluation dataset

Strategy

- Use purely LLM-based label-free evaluation
- Leverage LLM to generate synthetic label dataset, including
  - Query
  - “Golden” context
  - “Golden” response

Generating Synthetic Evaluation Data

- Sample document chunks, and leverage LLM to generate Q&A pairs that can be best answer by the given context.
- Not always representative of user queries, but useful for development iterations & diagnostics



RAY

The development process

At this stage

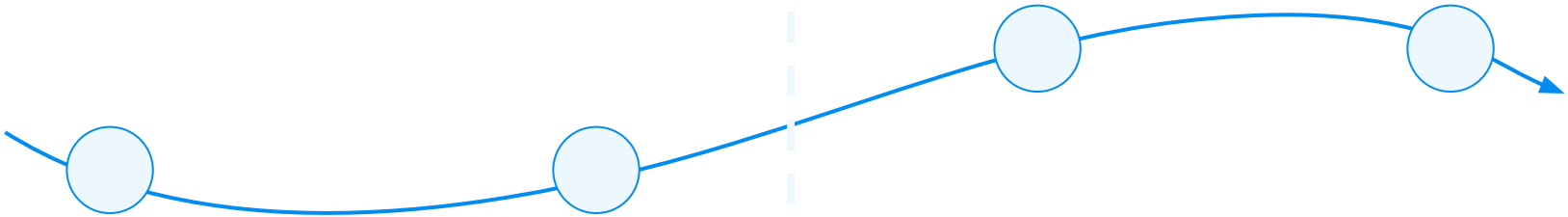
- Have user queries, maybe labels & feedback
- Need repeatable, consistent, automated evaluation
- Need actionable insight or automated tuning

Deployment

Deploy system & collect user queries & feedback

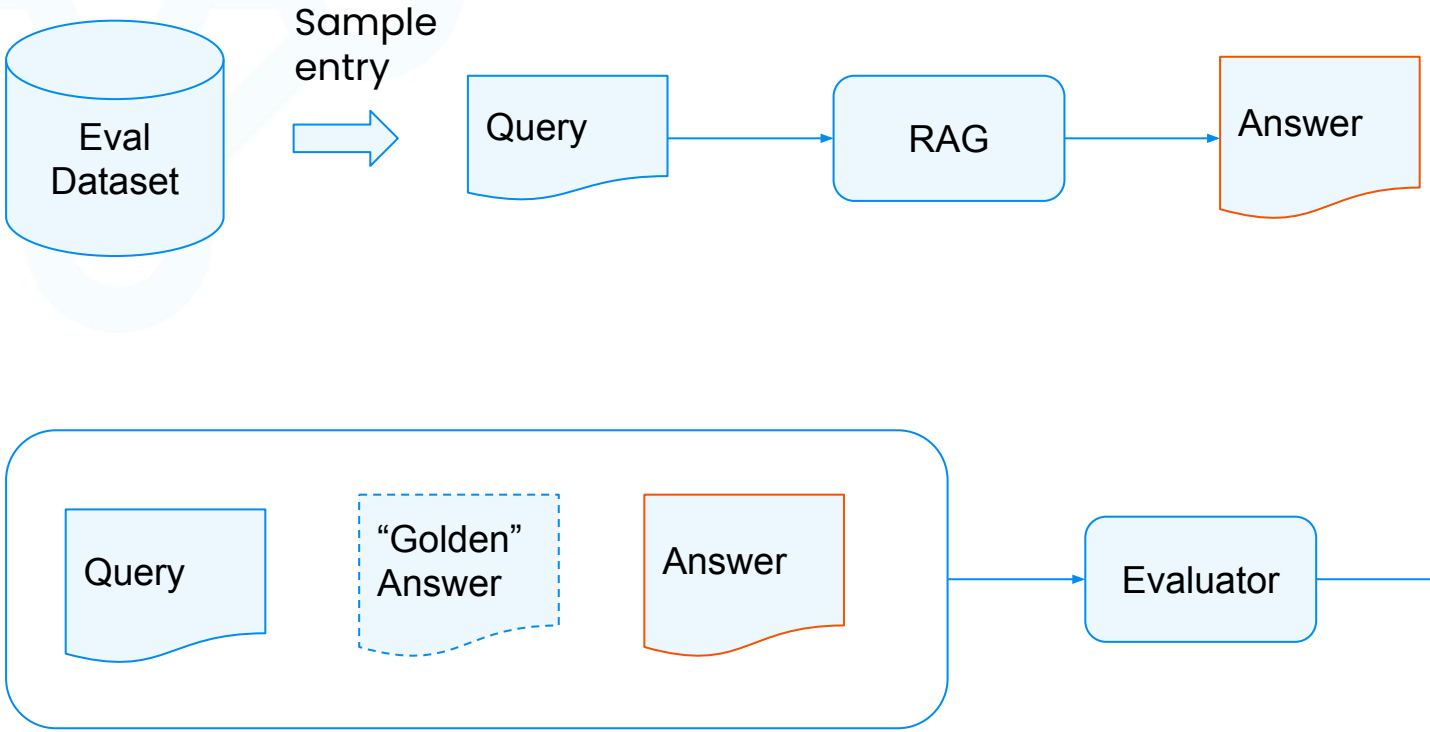
Optimization

Collect labels, improve system



RAY

End-to-end evaluation



RAY

LLM-as-a-Judge:
The devil is in the details

- **What model?**
  - Only “strong” LLMs right now
  - GPT-4, Claude 2
- **What grading scale?**
  - Low-precision, e.g. binary, 1-5
  - Easier rubrics, more interpretable
- **Do we need few-shot examples?**
  - Helps to give guidance/example for each score level
- **Holistic judgement vs. individual aspects**
  - Be as concrete as possible
- **Chain of thought reasoning**

```
You are an expert evaluation system for a question answering
chatbot.

You are given the following information:
- a user query,
- a reference answer, and
- a generated answer.

Your job is to judge the relevance and correctness of the
generated answer.
Output a single score that represents a holistic evaluation.
You must return your response in a line with only the score.
Do not return answers in any other format.
On a separate line provide your reasoning for the score as well.

Follow these guidelines for scoring:
- Your score has to be between 1 and 5, where 1 is the worst and 5
is the best.
- If the generated answer is not relevant to the user query, \
you should give a score of 1.
- If the generated answer is relevant but contains mistakes, \
you should give a score between 2 and 3.
- If the generated answer is relevant and fully correct, \
you should give a score between 4 and 5.

## User Query
{query}

## Reference Answer
{reference_answer}

## Generated Answer
{generated_answer}
```

RAY

LLM-as-a-Judge: Limitations

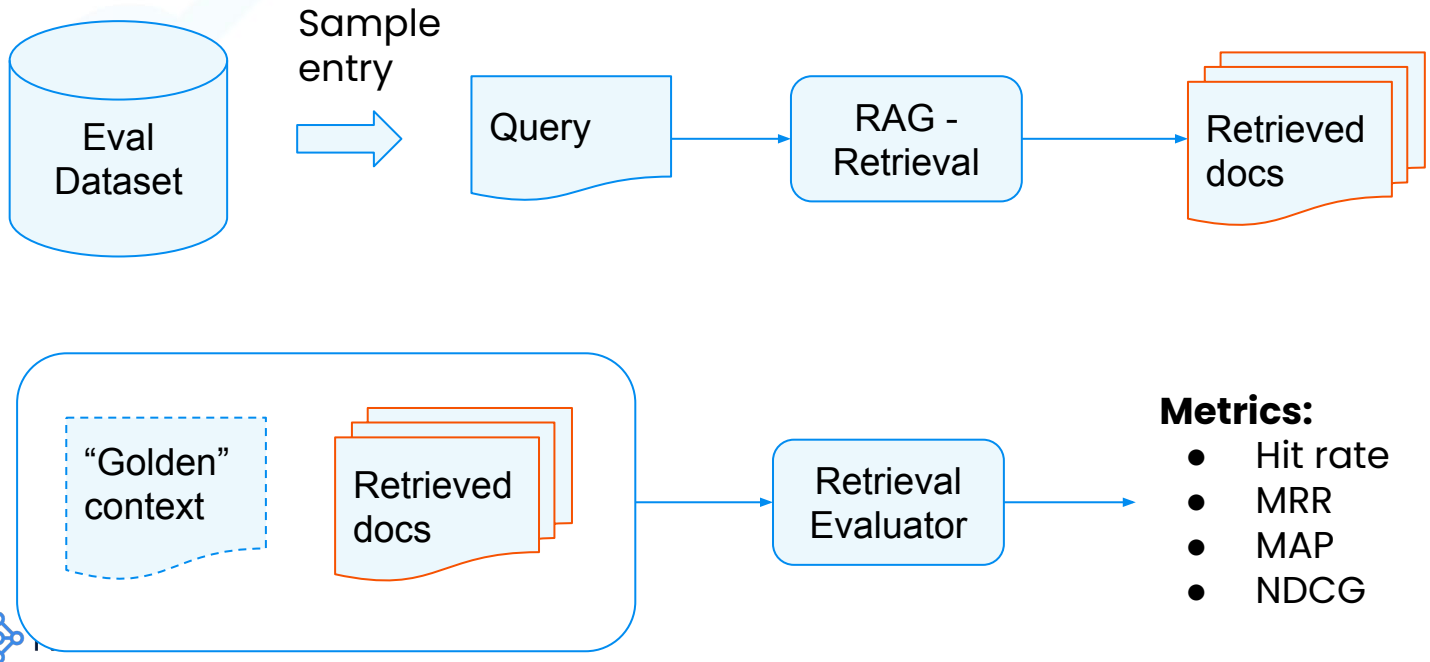
- **Position bias**
  - First, last, etc
- **Verbosity bias**
  - Longer the better?
- **Self-enhancement bias**
  - GPT <3 GPT

```
You are an expert evaluation system for a question answering
chatbot.

You are given the following information:
- a user query,
- a reference answer, and
- a generated answer.

Your job is to judge the relevance and correctness of the
generated answer.
Output a single score that represents a holistic evaluation.
You must return your response in a line with only the score.
Do not return answers in any other format.
On a separate line provide your reasoning for the score as well.

Follow these guidelines for scoring:
- Your score has to be between 1 and 5, where 1 is the worst and 5
is the best.
- If the generated answer is not relevant to the user query, \
you should give a score of 1.
- If the generated answer is relevant but contains mistakes, \
you should give a score between 2 and 3.
- If the generated answer is relevant and fully correct, \
you should give a score between 4 and 5.

## User Query
{query}

## Reference Answer
{reference_answer}

## Generated Answer
{generated_answer}
```

RAY

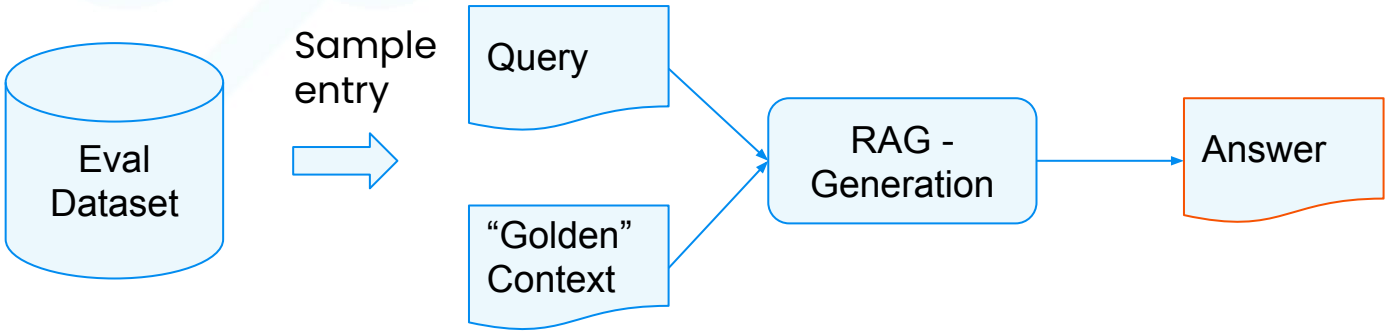
Component-wise evaluation

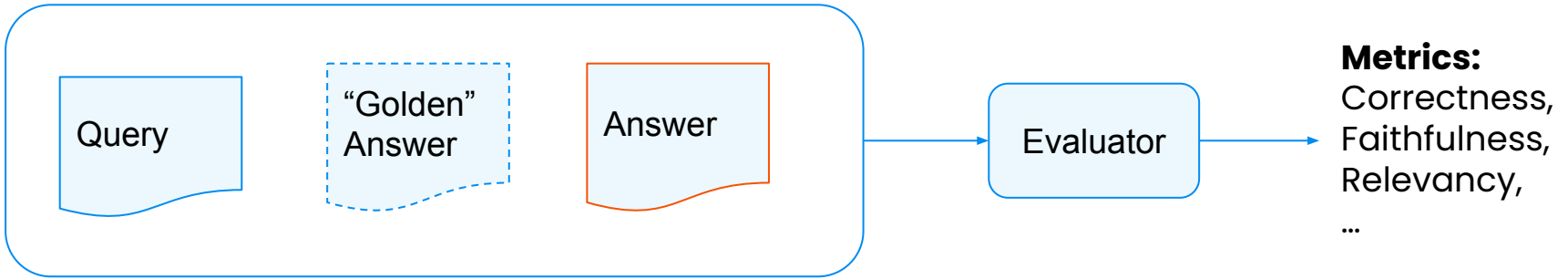
- Evaluate **retrieval** component

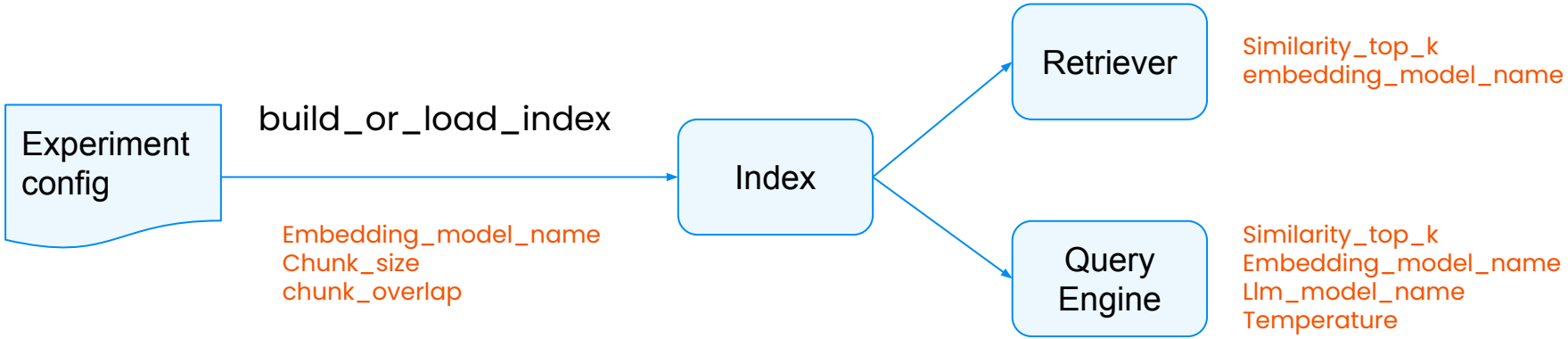


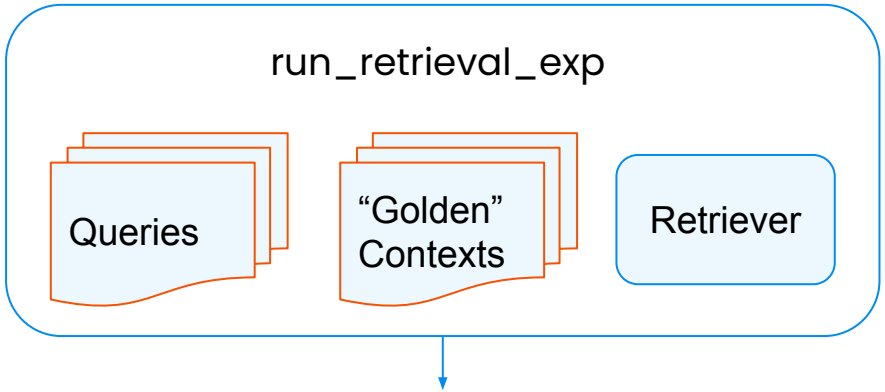
Component-wise evaluation

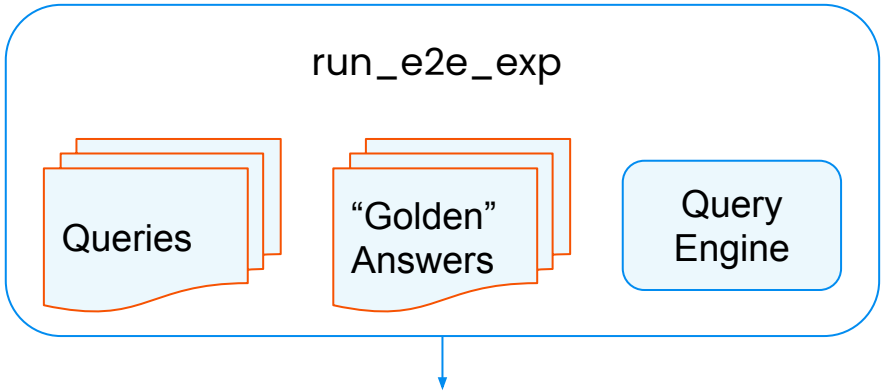
- Evaluate **generation** component






Experiment config
build_or_load_index
Embedding_model_name
Chunk_size
chunk_overlap
Index
Retriever
Similarity_top_k
embedding_model_name
Query Engine
Similarity_top_k
Embedding_model_name
Llm_model_name
Temperature


run_retrieval_exp
Queries
"Golden" Contexts
Retriever
Hit rate


run_e2e_exp
Queries
"Golden" Answers
Query Engine
Mean correctness score

Lab

- Cold start problem: generating synthetic eval dataset from documents
- Component wise evaluation
  - Retrieval
  - Generation
- End-to-end evaluation

Time for a Break!

15 minutes.

Experiment & Optimize

Improve system performance

Ways to optimize the RAG system

Configure standard components

- Use standard, off the shelf components (e.g. LLM, embedding model, standard semantic search)
- Select good components, tune parameters for end-performance

Build customized pipeline

- Deeply understand your data & query pattern
- define indexing & retrieval strategies optimized for your specific use-case

Model Fine-tuning

- Specialize embedding model and LLM to your domain

RAY

Configure standard components

Retrieval

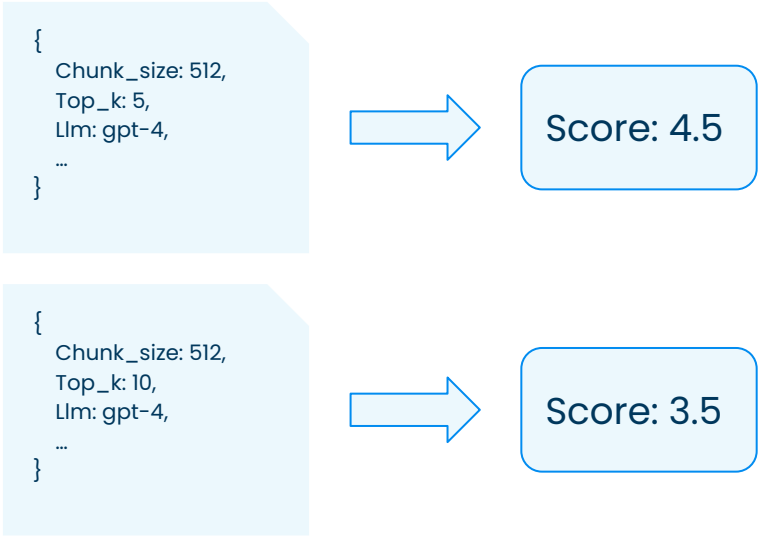
- Chunk size
- Embedding model
- Number of retrieved chunks

Generation

- LLM
- (prompt)

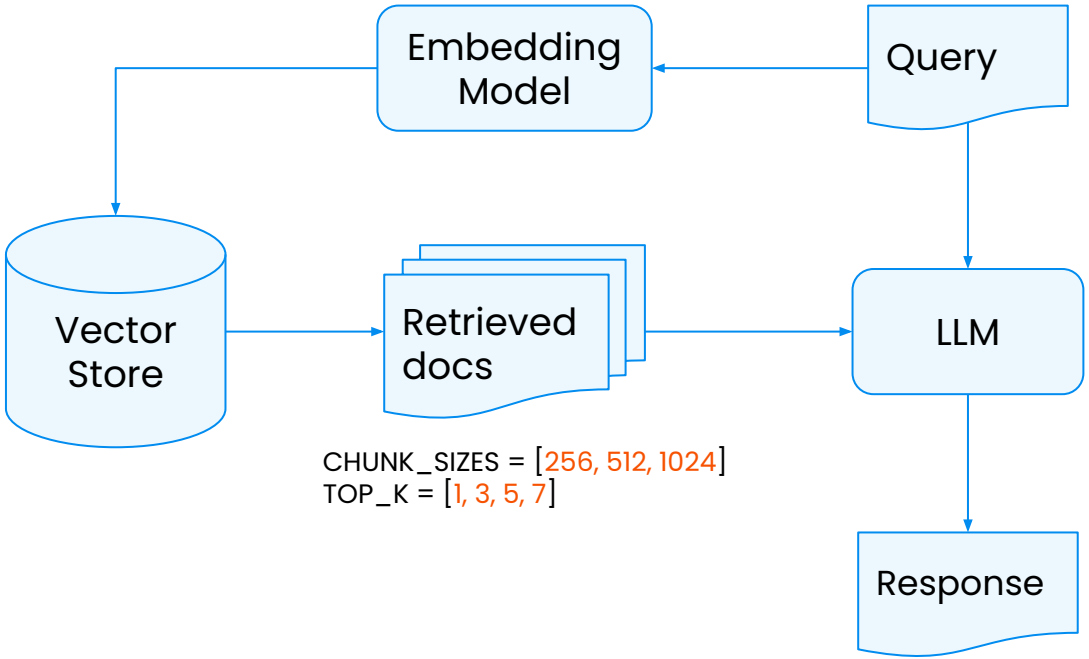
Grid search

- Run all parameter combinations and evaluate the end performance of system



RAY

```
EMBED_MODELS = [
    "thenlper/gte-base",
    "BAAI/bge-large-en",
    "text-embedding-ada-002"
]
```




```
LLMS = [
    "gpt-3.5-turbo",
    "gpt-4",
    "meta-llama/Llama-2-7b-chat-hf",
    "meta-llama/Llama-2-13b-chat-hf",
    "meta-llama/Llama-2-70b-chat-hf"
]
```

```
CHUNK_SIZES = [256, 512, 1024]
TOP_K = [1, 3, 5, 7]
```

Lab

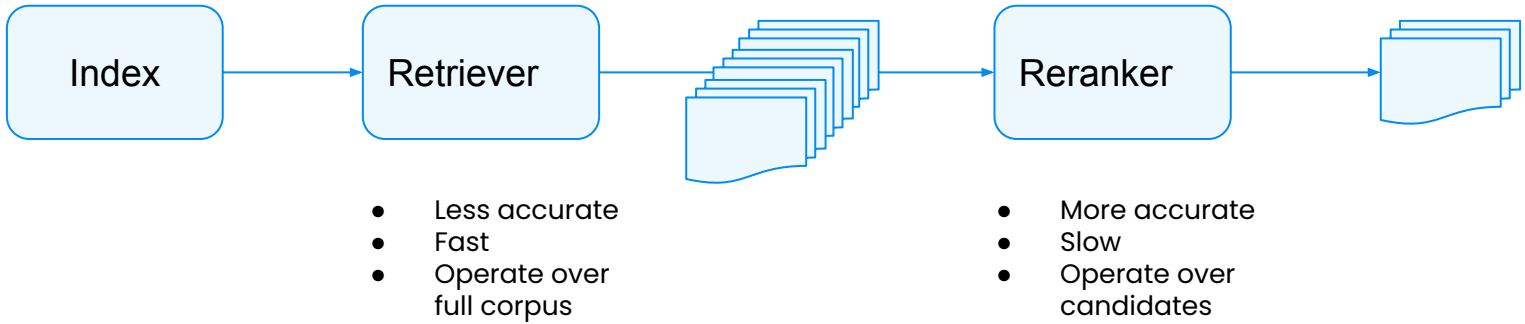
- Run experiments over standard component configurations
- Gain intuition on optimal parameter configurations

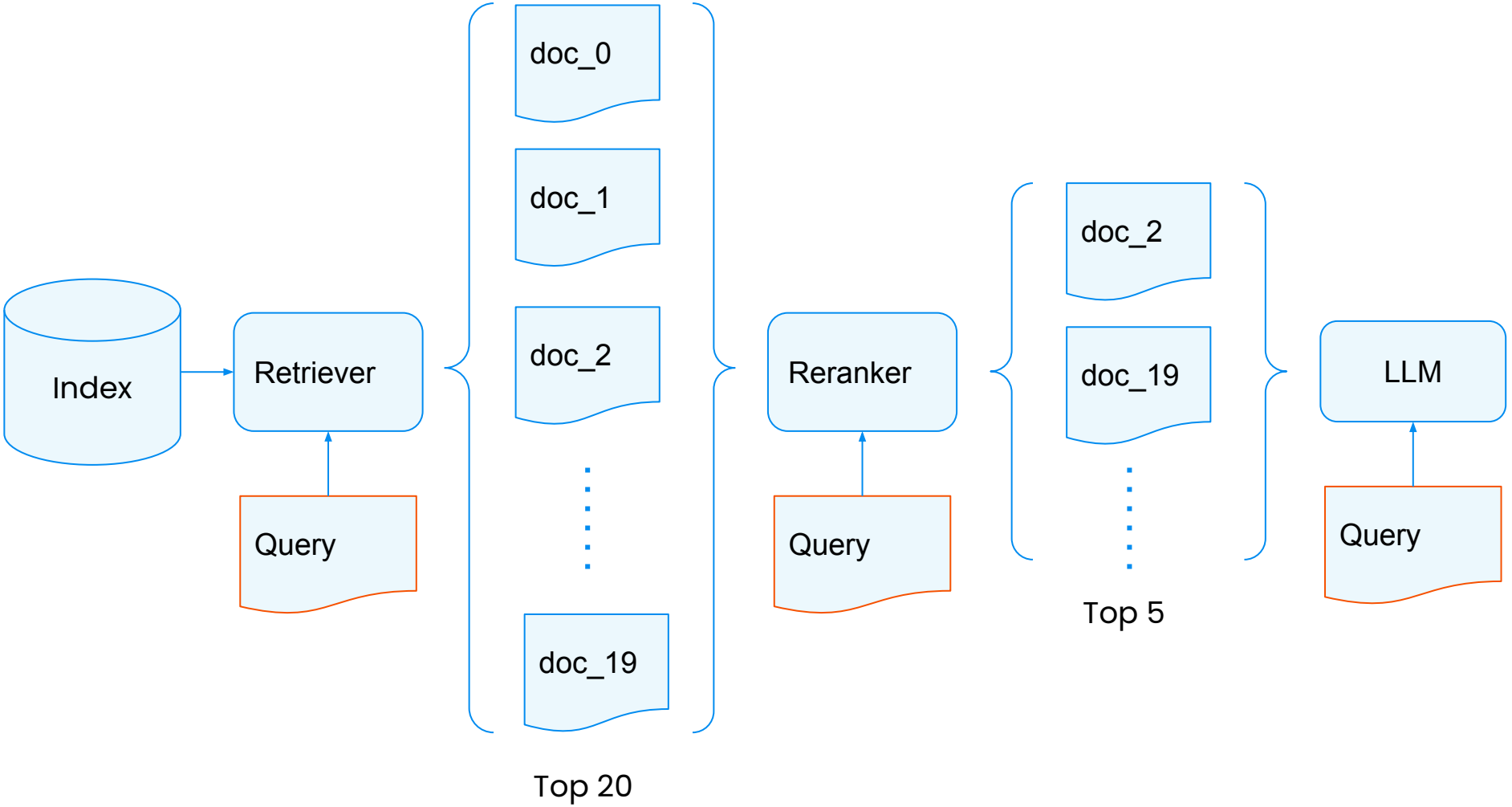
Customizing Retrieval & Generation

Strategy 1:

Use two-stage retrieval:

- First retrieve a lot of potentially relevant contexts
- Then rerank/filter to a smaller subset




Index
Retriever
Query
doc_0
doc_1
doc_2
doc_19
Top 20
Reranker
Query
doc_2
doc_19
Top 5
LLM
Query

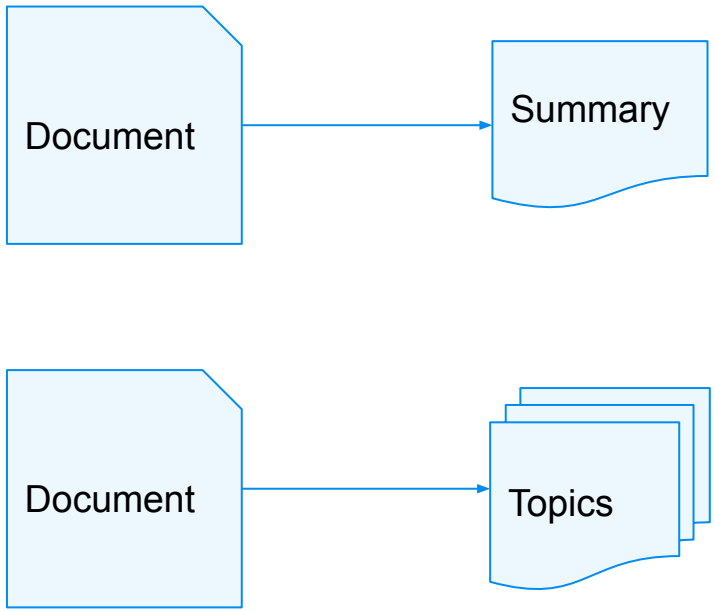
Customizing Retrieval & Generation

Strategy 2a:

Embed different representations of the same data: e.g.

- Summarize document and embed summary
- Extract distinct topics and embed topic extractions separately

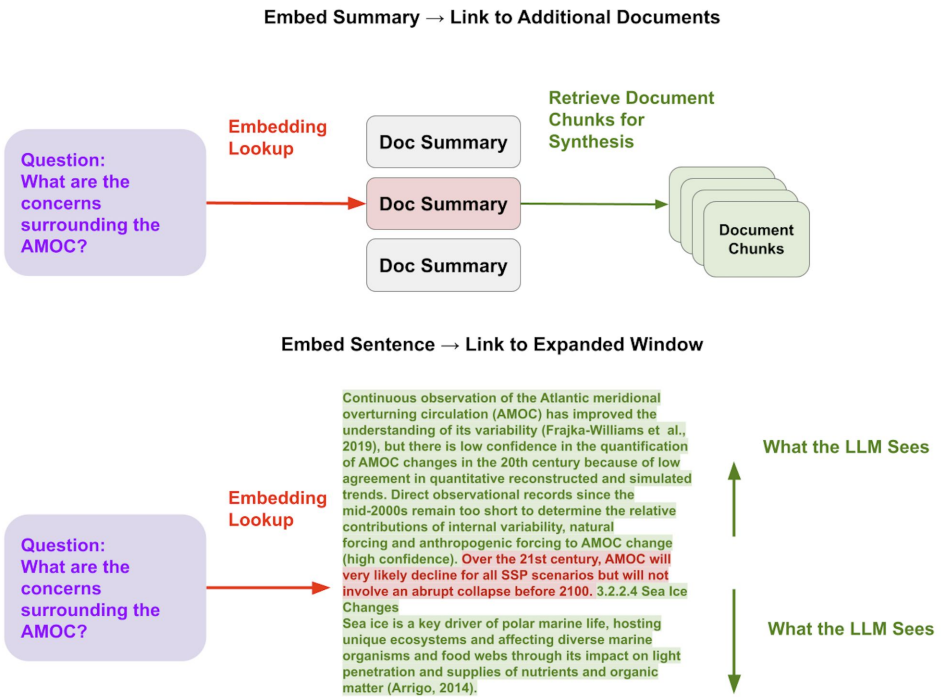
This can improve retrieval over specific questions



Customizing Retrieval & Generation

Strategy 2b:

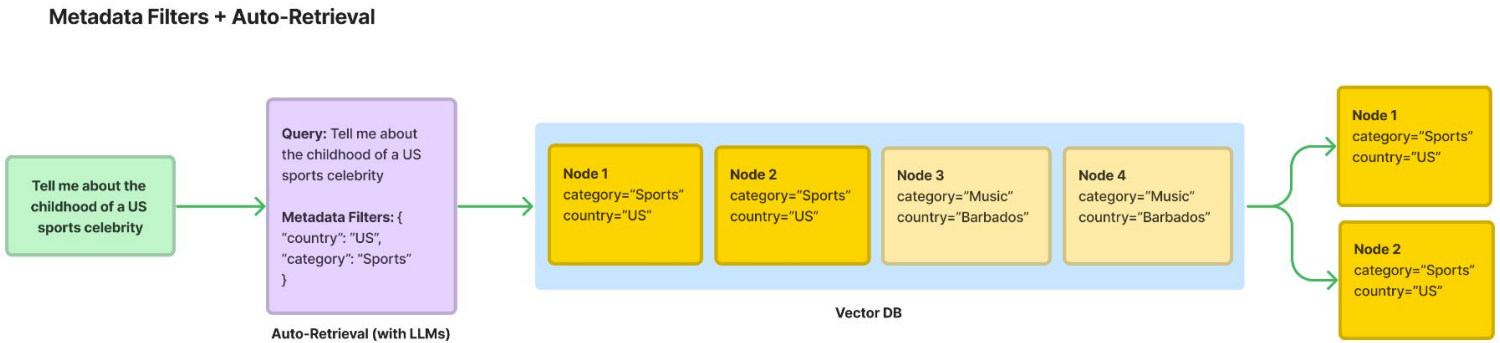
Use different data representation for retrieval & for generation



Customizing Retrieval & Generation

Strategy 3:

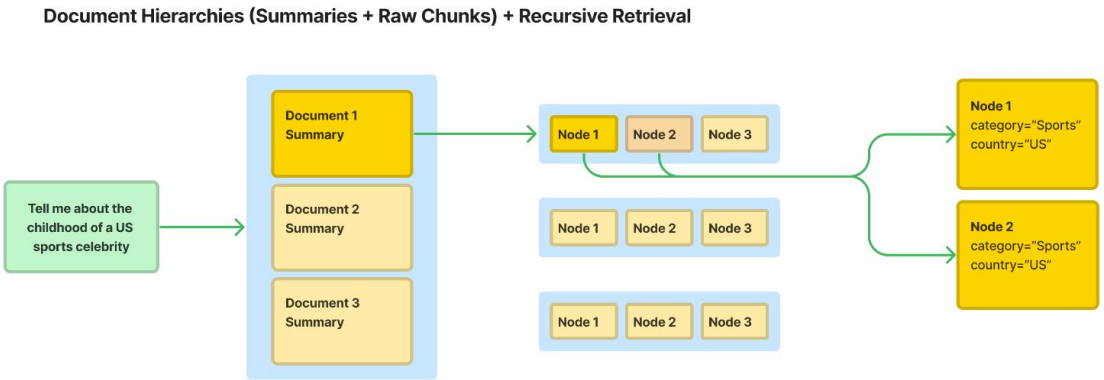
Leverage LLM to infer structured query for retrieval (e.g. metadata filters, top-k, score threshold)



Customizing Retrieval & Generation

Strategy 4:

Recursive retrieval over hierarchical index

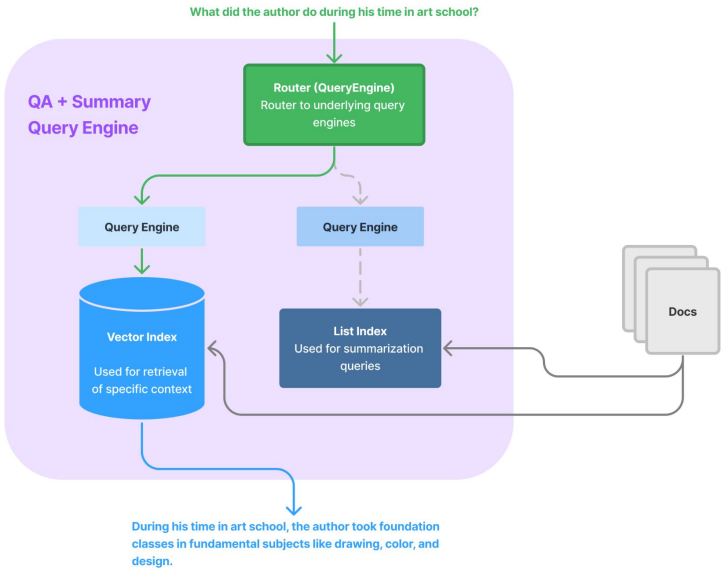


Customizing Retrieval & Generation

Strategy 5:

Routing to different retrieval methods depending on the query

May need to rewrite query as well, depending on the interface of the retriever.



Model Fine-tuning

LLM Fine-tuning

- When want to adjust the “style” of generation: e.g. professional legal assistant
- When want to enforce output structure: e.g. JSON

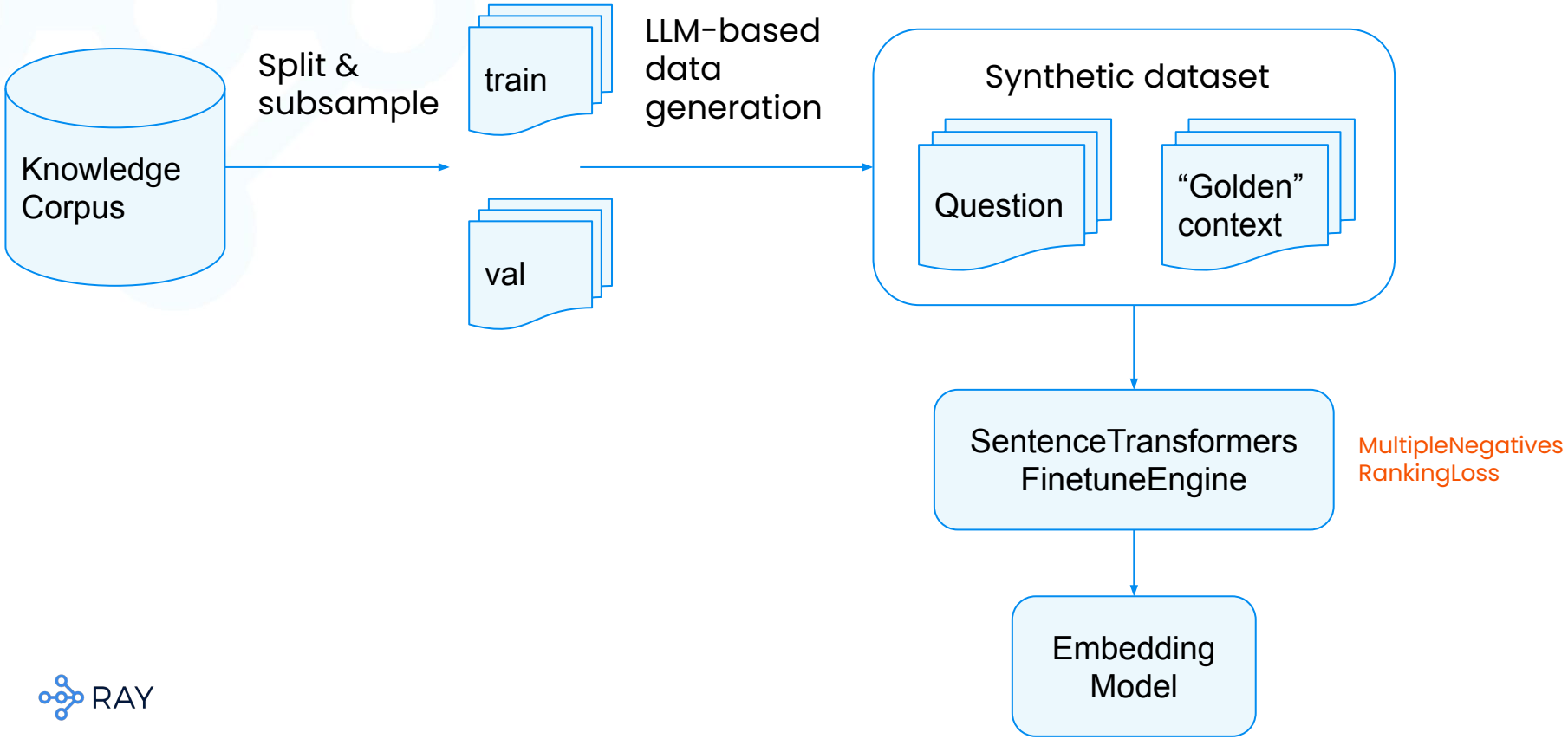
Not great for injecting new knowledge and fighting hallucination

Embedding Model Fine-tuning

- Improve retrieval performance, especially documents that are
  - Domain specific terminology
  - Malformed (e.g. extraneous spacing due to parsing, etc)
- Two approaches
  - Fine-tune the full embedding model
  - Fine-tune an adapter layer on top of a frozen embedding model

Great to improving retrieval (and thus end-to-end RAG performance)

Fine-tuning embeddings for RAG with synthetic data



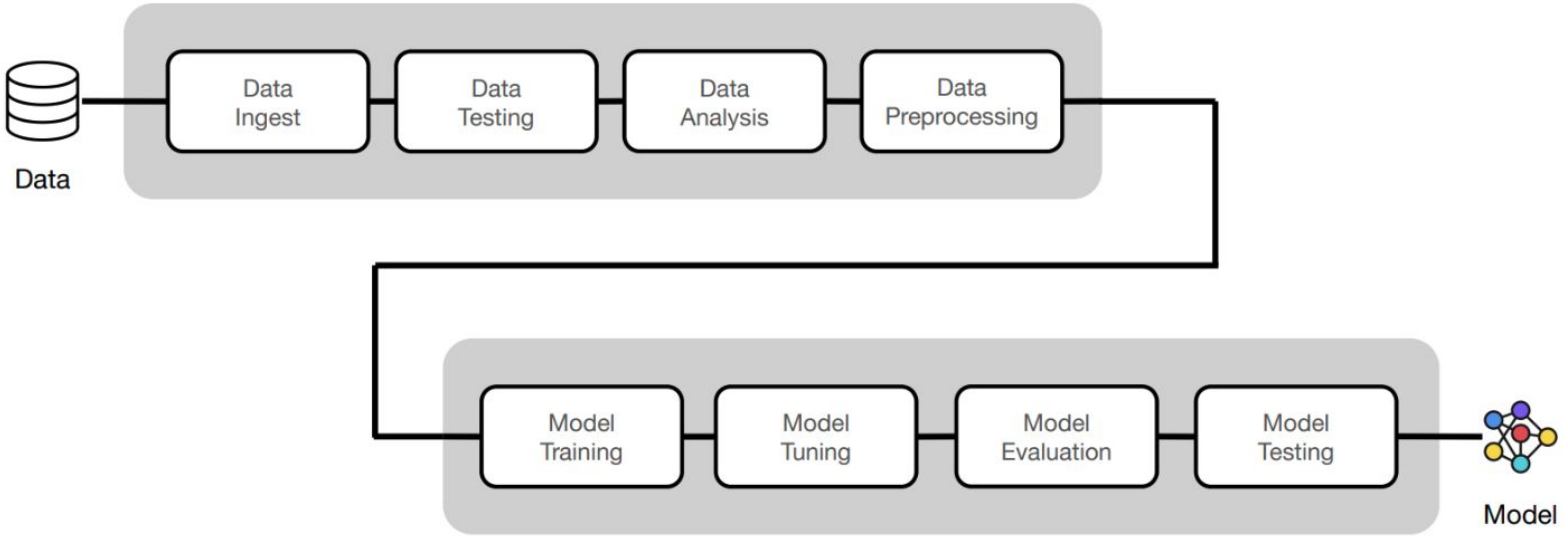
RAY

Lab

- Second-stage re-ranking
- Sentence window strategy
- Fine-tuning embeddings for RAG with synthetic data

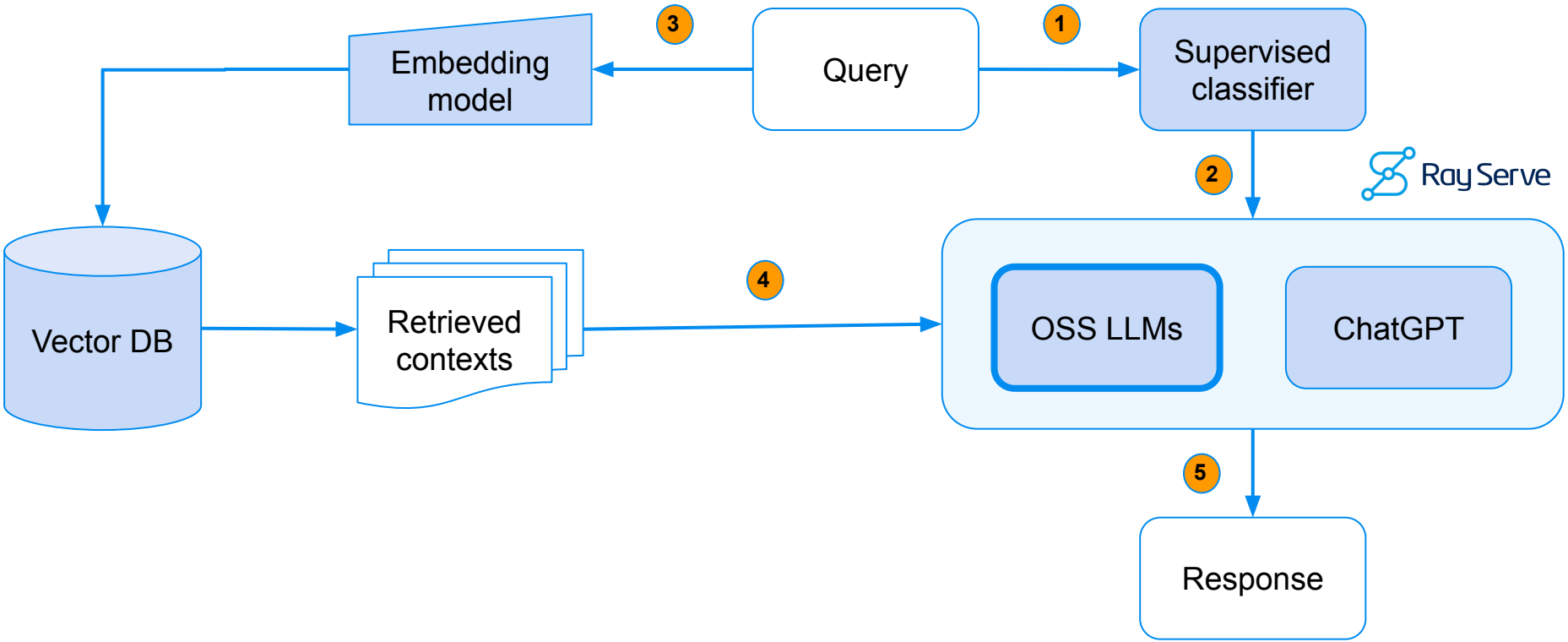
Deploy & Scale

Dev → Prod

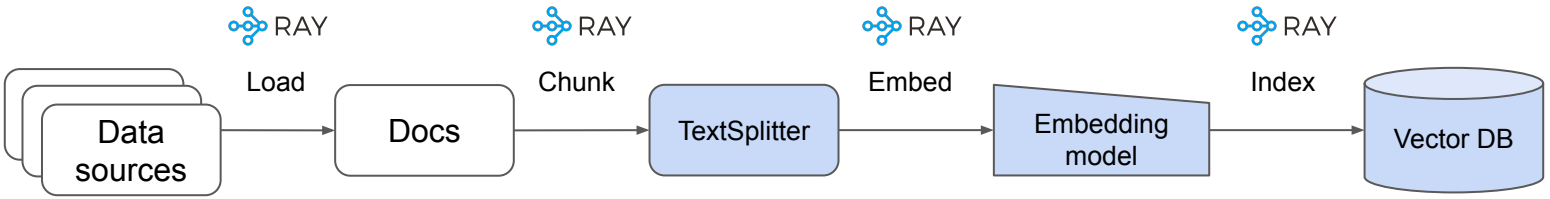



```
# ML workloads (simplified)
pytest --dataset-loc=$DATASET_LOC tests/data ...          # test data
python -m pytest tests/code --verbose --disable-warnings  # test code
python madewithml/train.py --experiment-name "llm" ...    # train model
python madewithml/evaluate.py --run-id $RUN_ID ...        # evaluate model
pytest --run-id=$RUN_ID tests/model ...                   # test model
python madewithml/serve.py --run_id $RUN_ID               # serve model
```

Hybrid routing

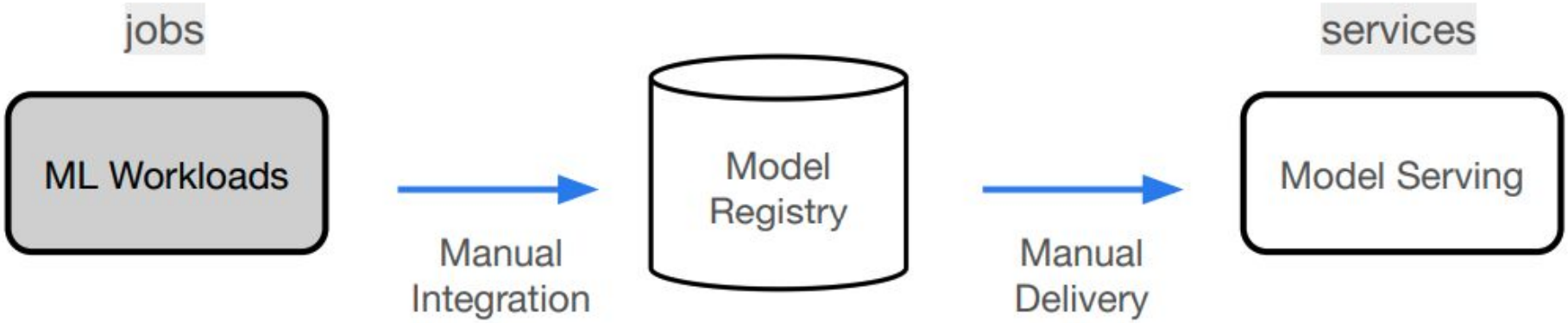


workloads.sh




```
# LLM workloads (simplified)
 pytest --dataset-loc=$DATASET_LOC tests/data ... # test data
 python -m pytest tests/code --verbose --disable-warnings # test code
 python rag/index.py --chunk-length 500 --embedding-model ... # create
index
 python rag/train.py --experiment-name "router" ... # train routing model
 python rag/evaluate.py --run-id $RUN_ID ... # evaluate routing model
 pytest --run-id=$RUN_ID tests/model ... # test model
 python rag/serve.py --run_id $RUN_ID # serve application
```

Production (manual)



workloads.yaml

```yaml
# deploy/jobs/workloads.yaml
name: workloads
project_id: prj_v9izs5t1d6b512ism8c5rkq4wm
cluster_env: rag-cluster-env
compute_config: rag-cluster-compute
entrypoint: bash deploy/jobs/workloads.sh
max_retries: 0
```

Jobs

```
anyscale job submit deploy/jobs/workloads.yaml
```

```
Authenticating

Output
(anyscale +8.8s) Maximum uptime is disabled for clusters launched by tl
(anyscale +8.8s) Job prodjob_zqj3k99va8a5jtd895u3ygraup has been succe
(anyscale +8.8s) Query the status of the job with `anyscale job list -
(anyscale +8.8s) Get the logs for the job with `anyscale job logs --jol
(anyscale +8.8s) View the job in the UI at https://console.anyscale.co
(anyscale +8.8s) Use --follow to stream the output of the job when subi
```

Duplicate as workspace | Archive | Dashboard

About this job

Status	ID	Created by
Success	prodjob_283ccv7jisw3uf6gq65nul…	goku+madewithml@anyscale.com
Access	**Project**	**Schedule**
Everyone in your organization can vie…	madewithml	-

Configuration

Runtime environment	Cluster environment	Compute config
View	madewithml-cluster-env:1	madewithml-cluster-compute
Cloud	**Entrypoint**	**Job config YAML**
madewithml-us-east-2	bash deploy/jobs/workloads.sh	View

Logs

Logs from latest job attempt | Ray logs

serve_rag.py

```python
# deploy/services/serve_model.py

import os
from rag.serve import RayAssistantDeployment

# Entrypoint
run_id = [line.strip() for line in open("run_id.txt")][0]
entrypoint = RayAssistantDeployment.bind(
    chunk_size=500,
    chunk_overlap=50,
    num_chunks=7,
    embedding_model_name="thenlper/gte-base",
    llm="meta-llama/Llama-2-70b-chat-hf")
```

```python
# Inference
data = {"query": "What is the default batch size for map_batches?"}
response = requests.post("http://127.0.0.1:8000/query", json=data)
print(response.json())
```

serve.yaml

```yaml
# deploy/services/serve_model.yaml
name: rag
project_id: prj_v9izs5t1d6b512ism8c5rkq4wm
cluster_env: rag-cluster-env
compute_config: rag-cluster-compute
ray_serve_config:
    import_path: deploy.services.serve_rag:entrypoint
rollout_strategy: ROLLOUT # ROLLOUT or IN_PLACE
```

Scaling up

```
# Compute config
 export CLUSTER_COMPUTE_NAME="rag-cluster-compute-prod"
 anyscale cluster-compute create deploy/cluster_compute_prod.yaml --name
$CLUSTER_COMPUTE_NAME # uses new config with prod compute requirements
```

```
name: madewithml
project_id: prj_v9izs5t1d6b512ism8c5rkq4wm
cluster_env: madewithml-cluster-env
compute_config:
  cloud: anyscale-v2-cloud-fast-startup
  max_workers: 20
  head_node_type:
    name: head_node_type
    instance_type: m5.4xlarge
  worker_node_types:
  - name: gpu_worker
    instance_type: g4dn.4xlarge
    min_workers: 1
    max_workers: 8
  aws:
    BlockDeviceMappings:
    - DeviceName: "/dev/sda1"
      Ebs:
        VolumeSize: 500
        DeleteOnTermination: true
...
```

Services

```
# Rollout service
anyscale service rollout -f deploy/services/serve_model.yaml
```

```
    Authenticating

    Output
    (anyscale +7.3s) Service service2_xwmyv1wcm3i7qan2sahsmybymw has been
    (anyscale +7.3s) View the service in the UI at https://console.anyscal
```

```
# Rollback (to previous version of the Service)
 anyscale service rollback -f $SERVICE_CONFIG --name $SERVICE_NAME

# Terminate
 anyscale service terminate --name $SERVICE_NAME
```

POST request

```
# Query
curl -X POST -H "Content-Type: application/json" -H "Authorization: Bearer
$SECRET_TOKEN" -d '{ "query": "What is the default size for map_batches?"
}' $SERVICE_ENDPOINT/predict/
```


```
{
    'question': 'What is the default batch size for map_batches?',
    'sources':
    ['https://docs.ray.io/en/master/data/api/doc/ray.data.Dataset.map_batches.htm
    l#ray-data-dataset-map-batches',
    ...
    'https://docs.ray.io/en/master/data/examples/huggingface_vit_batch_prediction
    .html#step-3-scaling-up-to-the-full-dataset-with-ray-data'],
    'answer': 'The default batch size for map_batches is 4096.',
    'Llm': 'meta-llama/Llama-2-70b-chat-hf'
}
```

Observability

Application name	Route prefix	Status	Status message	Num deployments	Last deployed at	Duration (since last deploy)	Application config
default	/	RUNNING	-	1	2023/07/24 13:31:50	7m 22s	View

Logs

CONTROLLER LOGS | OTHER LOGS

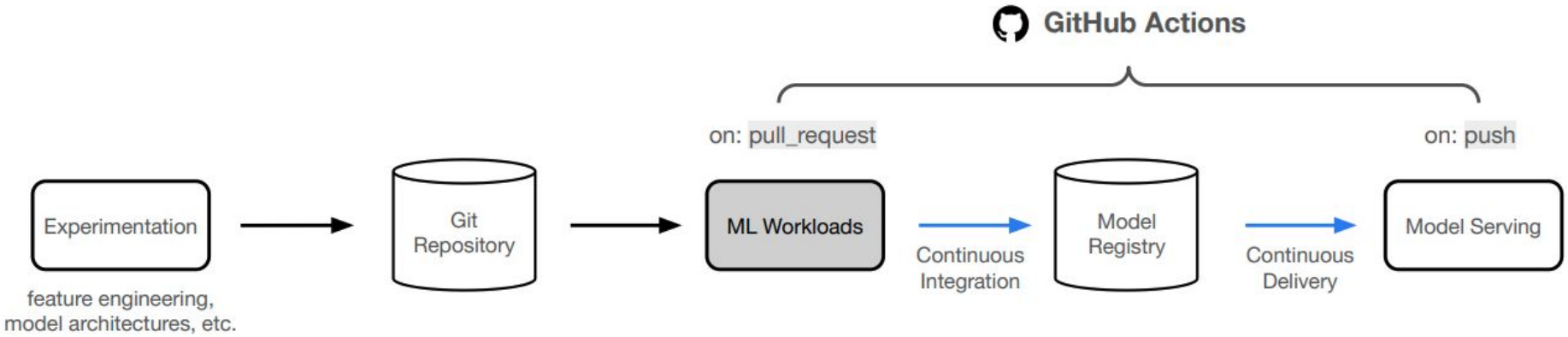
Keyword | Line Number | Font Size | Start Time mm/dd/yyyy, --:-- -- | End Time mm/dd/yyyy, --:-- --

Reverse: REFRESH RESET TIME DOWNLOAD LOG FILE

```
' http_state.py:436 - Starting HTTP proxy with name 'SERVE_CONTROLLER_ACTOR:SERVE_PROXY_ACTOR-ddaff3a3a7072bbccdeff7d7a82206
' http_state.py:436 - Starting HTTP proxy with name 'SERVE_CONTROLLER_ACTOR:SERVE_PROXY_ACTOR-5a2ae045768d27b86db49027985387
' controller.py:670 - Starting deploy_serve_application task for application default.
' deployment_state.py:1308 - Deploying new version of deployment default_ModelDeployment.
' deployment_state.py:1571 - Adding 1 replica to deployment default_ModelDeployment.
' deployment_state.py:353 - Starting replica default_ModelDeployment#XFpHzv for deployment default_ModelDeployment.
' application_state.py:356 - Deploy task for app 'default' ran successfully.
' deployment_state.py:1725 - Replica default_ModelDeployment#XFpHzv started successfully on node 5a2ae045768d27b86db49027985
```

Observability

Production (CI/CD)



Production (CI/CD)

```yaml
# .github/workflows/workloads.yaml
name: workloads
on:
  workflow_dispatch:  # manual
  pull_request:
    branches:
    - main
...
```

```yaml
# Run workloads
- name: Workloads
run: |
    export ANYSCALE_HOST=$\{\{ secrets.ANYSCALE_HOST \}\}
    export ANYSCALE_CLI_TOKEN=$\{\{ secrets.ANYSCALE_CLI_TOKEN \}\}
    anyscale jobs submit deploy/jobs/workloads.yaml --wait
```

Production (CI/CD)

```yaml
# Comment results to PR
- name: Comment training results on PR
uses: thollander/actions-comment-pull-request@v2
with:
    filePath: results/training_results.md
- name: Comment evaluation results on PR
uses: thollander/actions-comment-pull-request@v2
with:
    filePath: results/evaluation_results.md
```

Production (CI/CD)

github-actions bot commented 2 weeks ago

timestamp:

July 10, 2023 11:08:02 PM

run_id:

f28c284feae34312a6229b84f7b4675c

params:

Key	Value
dropout_p	0.5
lr	0.0
lr_factor	0.8
lr_patience	3
num_samples	None
num_epochs	10
batch_size	256
num_classes	4

metrics:

epoch	train_loss	val_loss
0	5.772e-01	4.937e-01
1	4.762e-01	4.175e-01
2	3.687e-01	3.735e-01
3	2.930e-01	2.755e-01
4	2.102e-01	2.391e-01
5	1.618e-01	1.977e-01
6	1.136e-01	1.599e-01
7	7.671e-02	1.742e-01
8	5.616e-02	1.525e-01
9	3.934e-02	1.604e-01

github-actions bot commented 2 weeks ago

timestamp:

July 10, 2023 11:08:23 PM

run_id:

f28c284feae34312a6229b84f7b4675c

overall:

Key	Value
precision	0.935
recall	0.932
f1	0.931
num_samples	191.0

per_class:

Key	Value
computer-vision	{'precision': 0.971, 'recall': 0.93, 'f1': 0.95, 'num_samples': 71.0}
natural-language-processing	{'precision': 0.895, 'recall': 0.987, 'f1': 0.939, 'num_samples': 78.0}
other	{'precision': 0.92, 'recall': 0.885, 'f1': 0.902, 'num_samples': 26.0}
mlops	{'precision': 1.0, 'recall': 0.75, 'f1': 0.857, 'num_samples': 16.0}

slices:

Key	Value
nlp_llm	{'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'num_samples': 28}
short_text	{'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'num_samples': 7}

LLM application PRs?

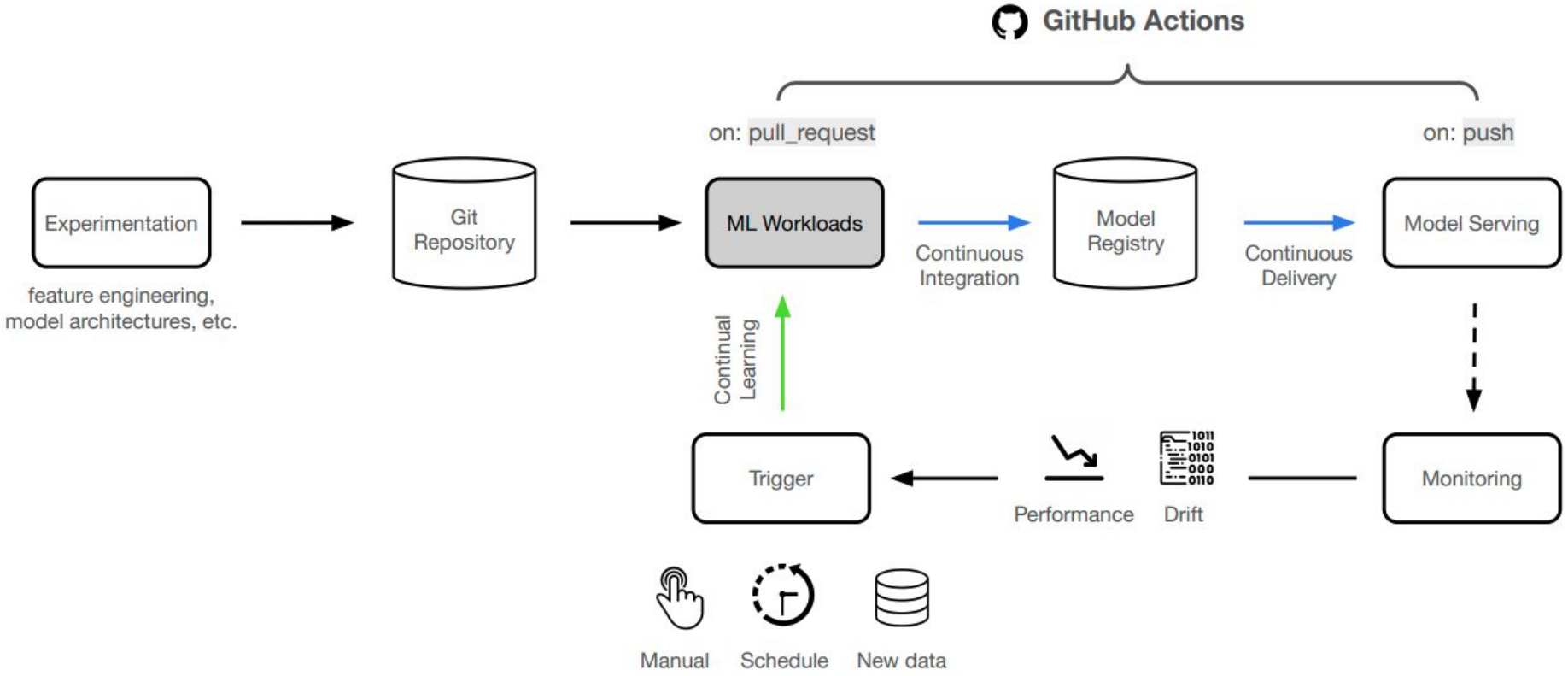
- LLM application config (chunking, embedding model, etc.)
- Retrieval and performance scores
- Retrieval and performance scores from current production deployment
- Link to data sources used
- High quality responses generated for simple test set
- Routing behavior for simple test set

Production (CI/CD)

```yaml
# .github/workflows/serve.yaml
name: serve
on:
  workflow_dispatch:  # manual
  push:
    branches:
    - main
...
```

```yaml
# Run workloads
- name: Workloads
run: |
    export ANYSCALE_HOST=$\{\{ secrets.ANYSCALE_HOST \}\}
    export ANYSCALE_CLI_TOKEN=$\{\{ secrets.ANYSCALE_CLI_TOKEN \}\}
    anyscale service rollout --service-config-file deploy/services/serve_m
```

Continual learning



Monitoring LLM applications?

- Spikes in negative user feedback
- Assess quality scores for small sample of production queries
- Scores for toxicity (queries and responses)
- Verbose responses
- Low similarity scores for top-k retrieved contexts
- Embedding clusters that are out of distribution

More Resources

For further exploration with Ray, Anyscale, and LLMs.

🍏 Sneak Peek: Self-Paced Ray & Anyscale Training

📧 Online at training.anyscale.com

🔍 Preview special technical content releases from the whole team!

Fill out the survey.

🔗 Go to bit.ly/ray-summit-feedback

Reading list.

Ray Education GitHub

Access bonus notebooks and scripts about Ray.

Ray documentation

API references and user guides.

Anyscale Blogs

Real world use cases and announcements.

YouTube Tutorials

Video walkthroughs about learning LLMs with Ray.

Upcoming events

Bay Area AI + Ray Summit Happy Hour

Today at 5:00p.m.

Cap off an exciting conference with lightning talks, new friends, and good times!

bit.ly/bayai_ray_meetup

Connect with the community.

👋 Join the community

Attend events, *subscribe to newsletter*, *follow on Twitter*.

🙋 Get support

Join Ray Slack, *ask questions on forum*, *open an issue*.

🔭 Contribute to Ray

Read contributor guide, *create a pull request*.

Thank you!

We hope to meet again.

Ray Summit 2023 Color Palette

Slide Template

Keynotes

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incid idunt ut labo re et dolore magna aliqu Ut enim ad minim veniam, quis nostrud exercitation

Here is an info card

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incid idunt ut labo re et dolore magna aliqu Ut enim ad minim veniam, quis nostrud exercitation

Slide Template

Keynotes

- Start to storyboard the keynote presentations
- Build out the stage design and presentation requirements
- Connect with external speakers on themes/topics

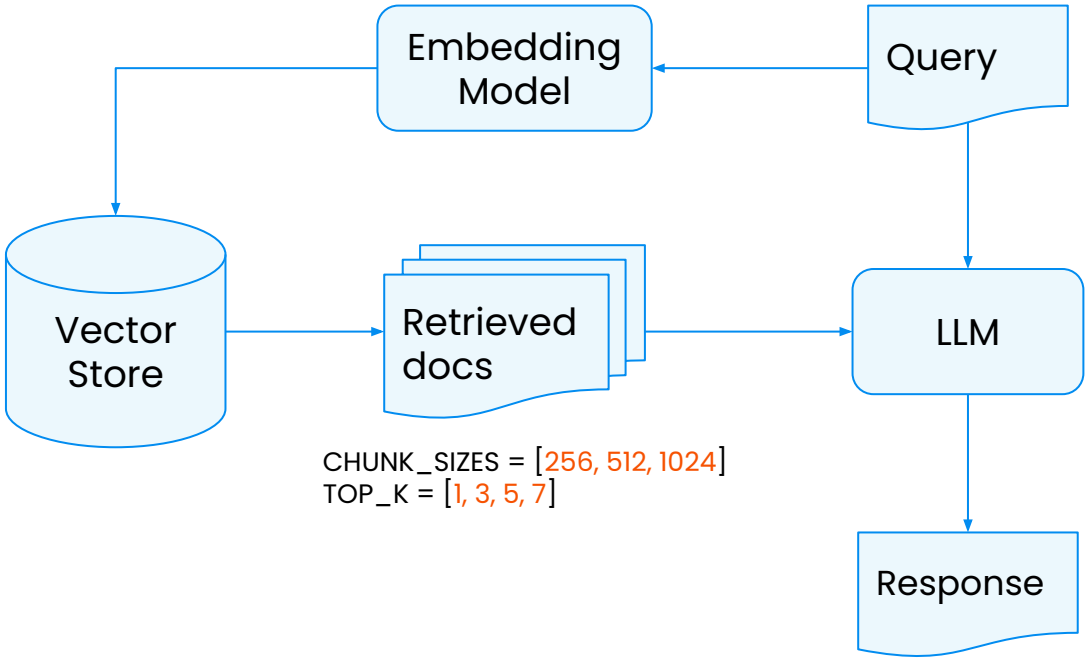
Production Costs

- Original estimates are lower than the quotes now coming in.
- Upgraded production quality results

Registration

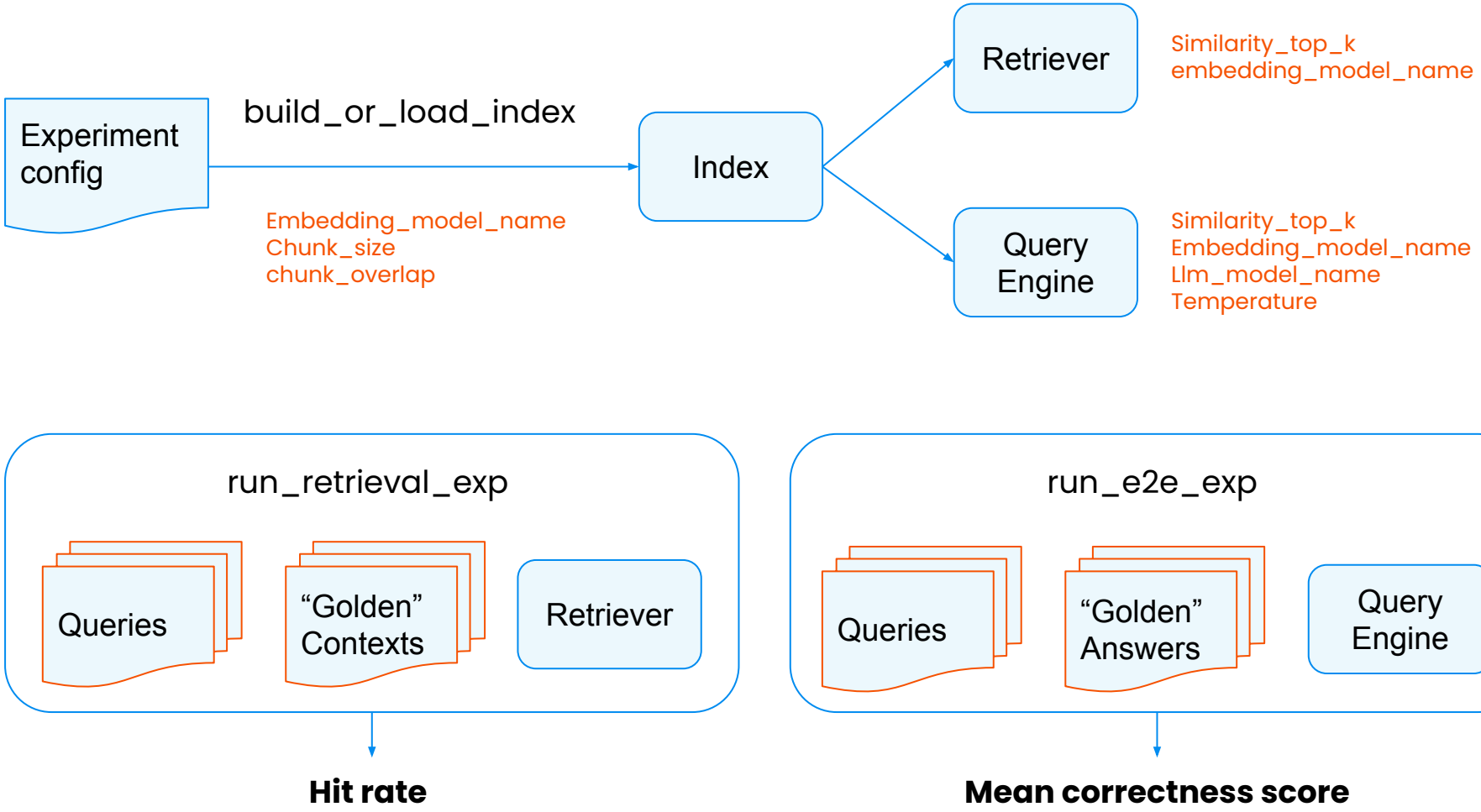
- Registration will continue to be a main area of focus especially as we approach

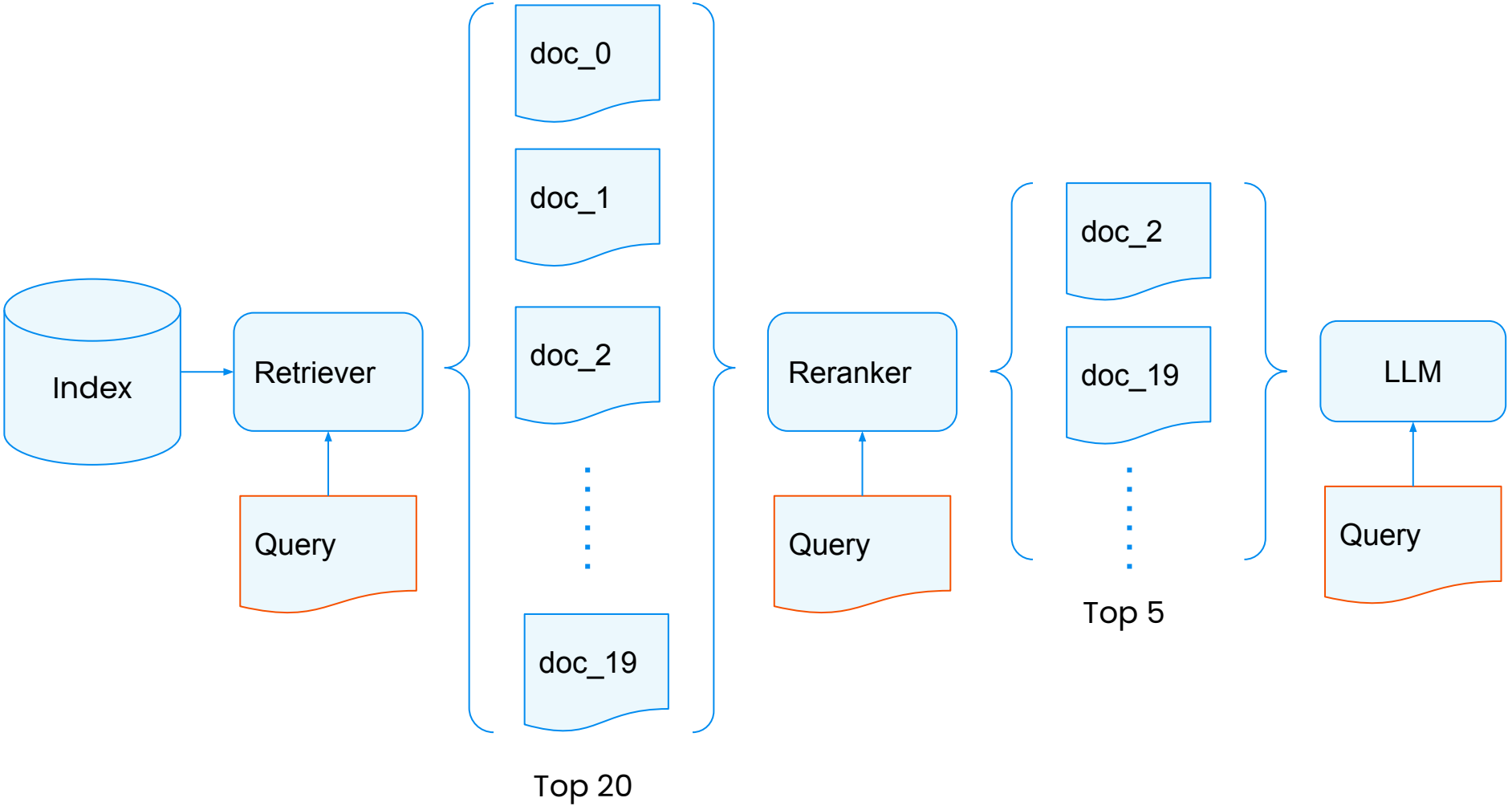
```
EMBED_MODELS = [
    "thenlper/gte-base",
    "BAAI/bge-large-en",
    "text-embedding-ada-002"
]
```

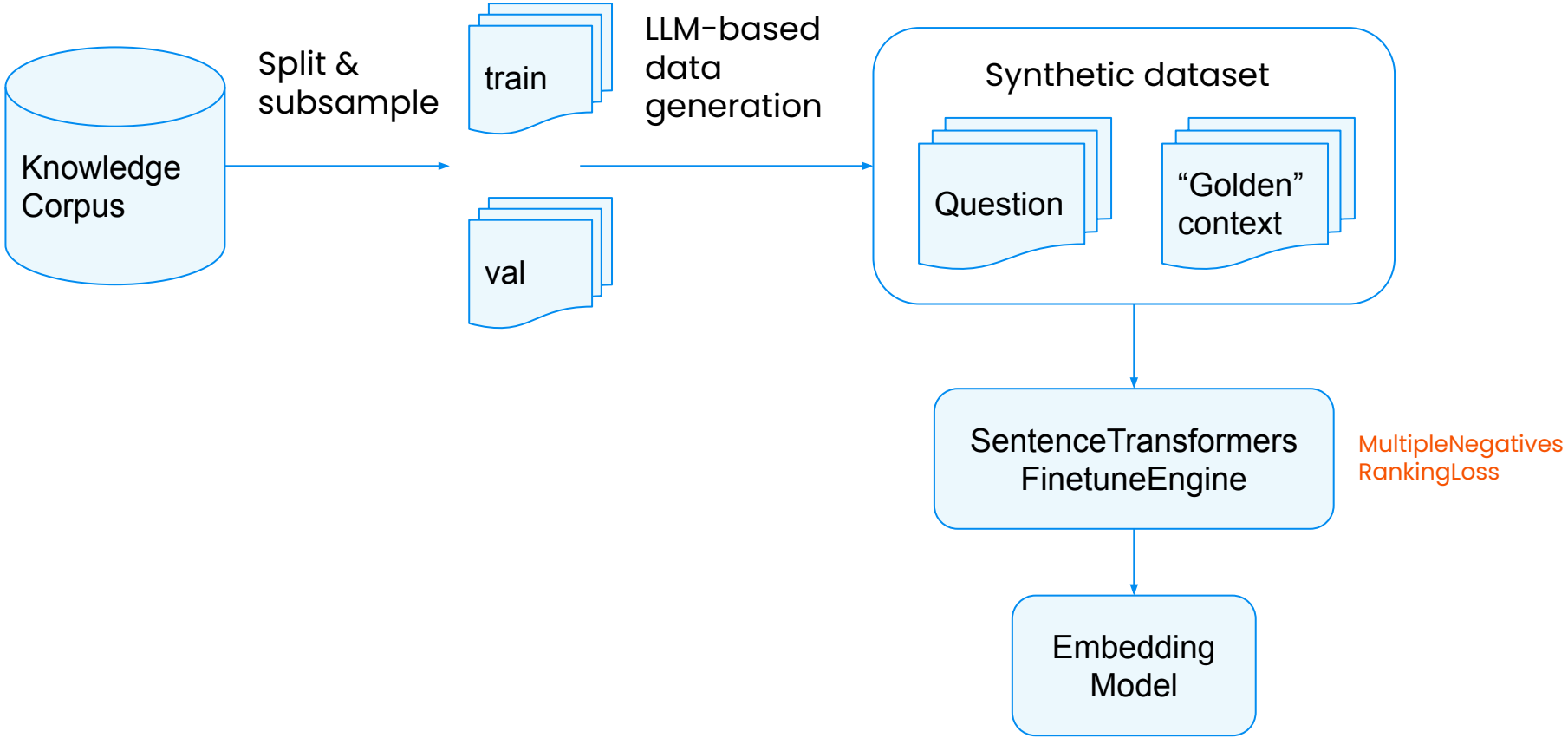



```
CHUNK_SIZES = [256, 512, 1024]
TOP_K = [1, 3, 5, 7]
```

```
LLMS = [
    "gpt-3.5-turbo",
    "gpt-4",
    "meta-llama/Llama-2-7b-chat-hf",
    "meta-llama/Llama-2-13b-chat-hf",
    "meta-llama/Llama-2-70b-chat-hf"
]
```


Experiment config
build_or_load_index
Embedding_model_name
Chunk_size
chunk_overlap
Index
Retriever
Similarity_top_k
embedding_model_name
Query Engine
Similarity_top_k
Embedding_model_name
Llm_model_name
Temperature
run_retrieval_exp
Queries
"Golden" Contexts
Retriever
Hit rate
run_e2e_exp
Queries
"Golden" Answers
Query Engine
Mean correctness score


Index
Retriever
Query
doc_0
doc_1
doc_2
doc_19
Top 20
Reranker
Query
doc_2
doc_19
Top 5
LLM
Query


Knowledge Corpus
Split & subsample
train
val
LLM-based data generation
Synthetic dataset
Question
“Golden” context
SentenceTransformers FinetuneEngine
MultipleNegatives RankingLoss
Embedding Model