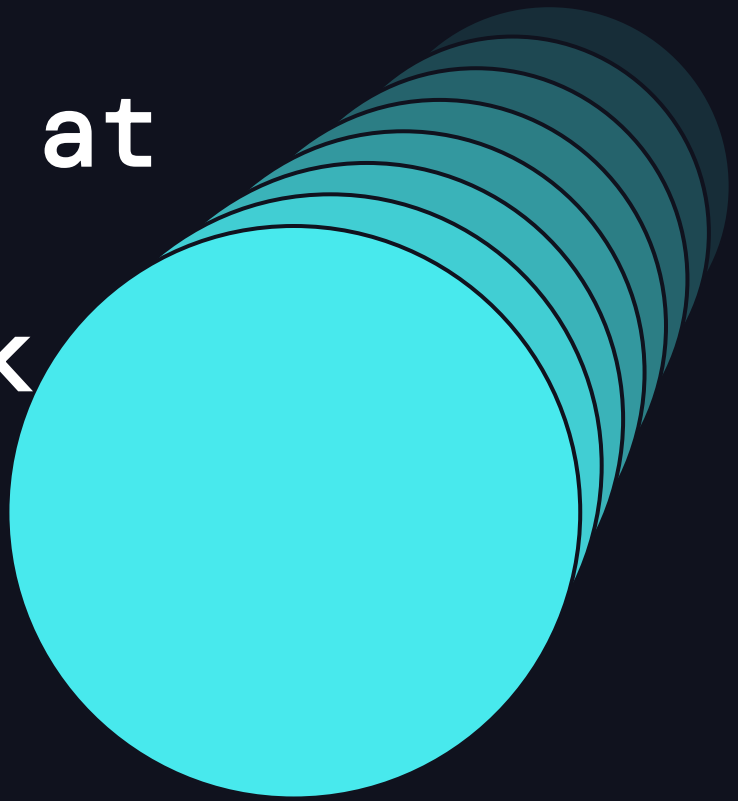
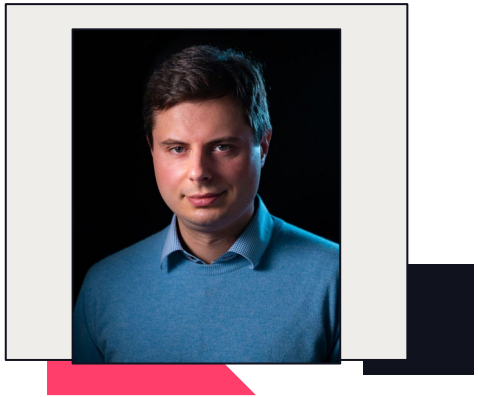
DATA+AI SUMMIT
BY databricks

Model Alignment at Scale using RL from AI Feedback on Databricks



Michael Shtelma
Ryuta Yoshimatsu
Alex Miller

Team



Michael Shtelma
Lead Specialist Solutions Architect at Databricks

Ryuta Yoshimatsu
Specialist Solutions Architect at Databricks

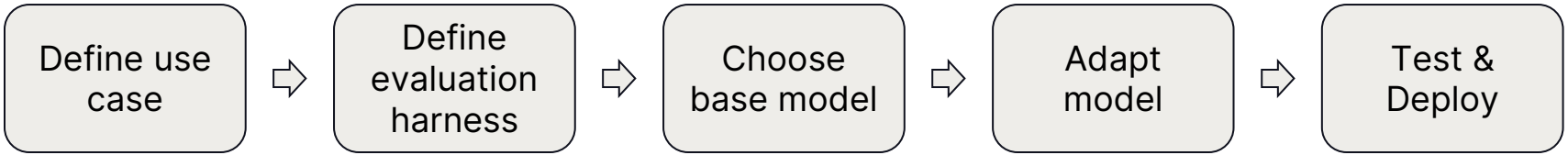
Alex Miller
Specialist Solutions Architect at Databricks

Agenda

- What is Model Alignment and why do we need it?
- Using RLHF to align models
- RLAIF: Using LLM as a Reward Model
- DPO: Direct Preference Optimization
- Model Alignment Solution Accelerator
- Implementation details
- Test Results
- Important Metrics

What is Model Alignment?

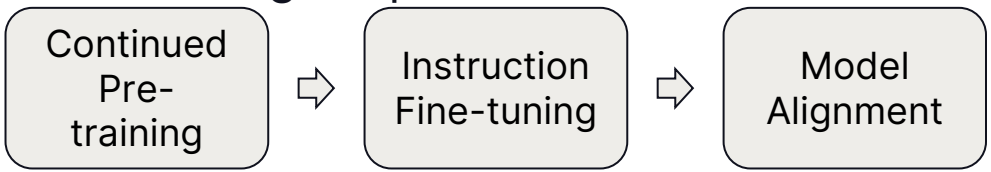
How does the typical LLM project looks like?



What is Model Alignment?

Model Adaptation phase

- Model Adaptation may include:
  - Continued pre-training
  - Supervised Instruction Fine-tuning
  - Model Alignment
- We need model alignment if it’s hard to express the business requirements using simple instructions



Using RLHF to align models

Reinforcement Learning from Human Feedback

- RLHF was introduced as a tool to align LLMs to human preferences by OpenAI in their InstructGPT paper (https://arxiv.org/pdf/2203.02155)
- RLHF usually consists of the following steps:
  - Instruction Fine-tuning
  - Training a Reward Model
  - RL optimization (often using Proximal Policy Optimization (PPO) algorithm)

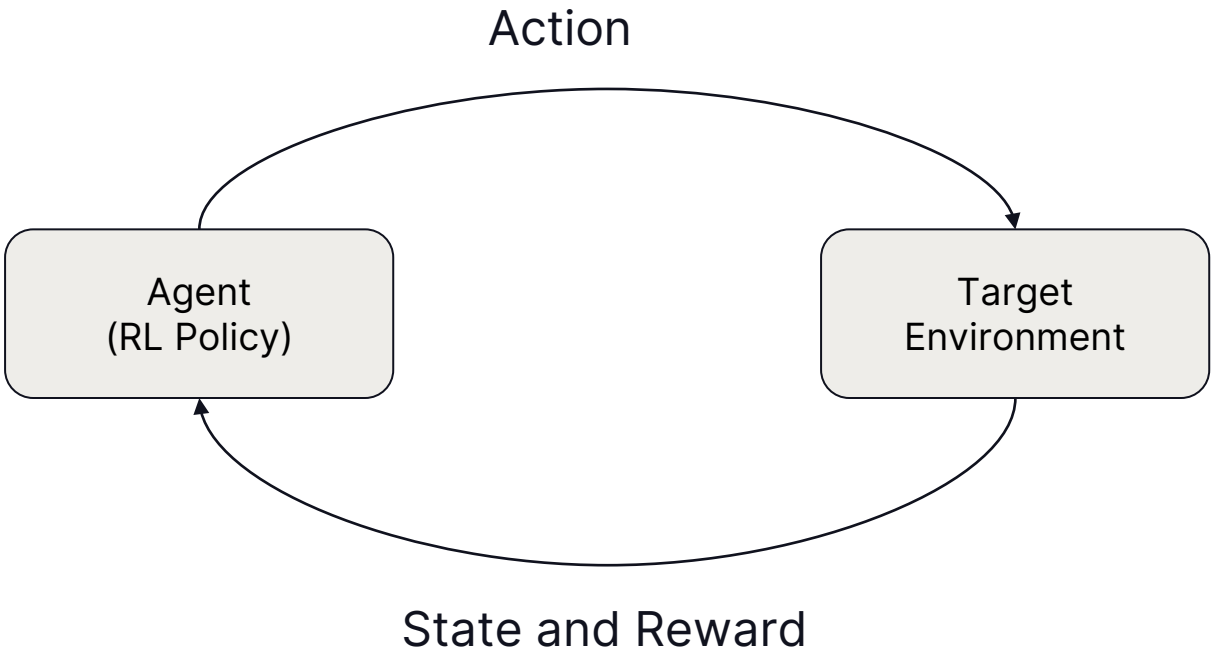
Using RLHF to align models

Reward Model

- Reward model is a classifier which provides a reward value for each question and model response
- We can use very different methods for training a reward model ranging from LLMs to complicated compound functions using multiple models and heuristic rules

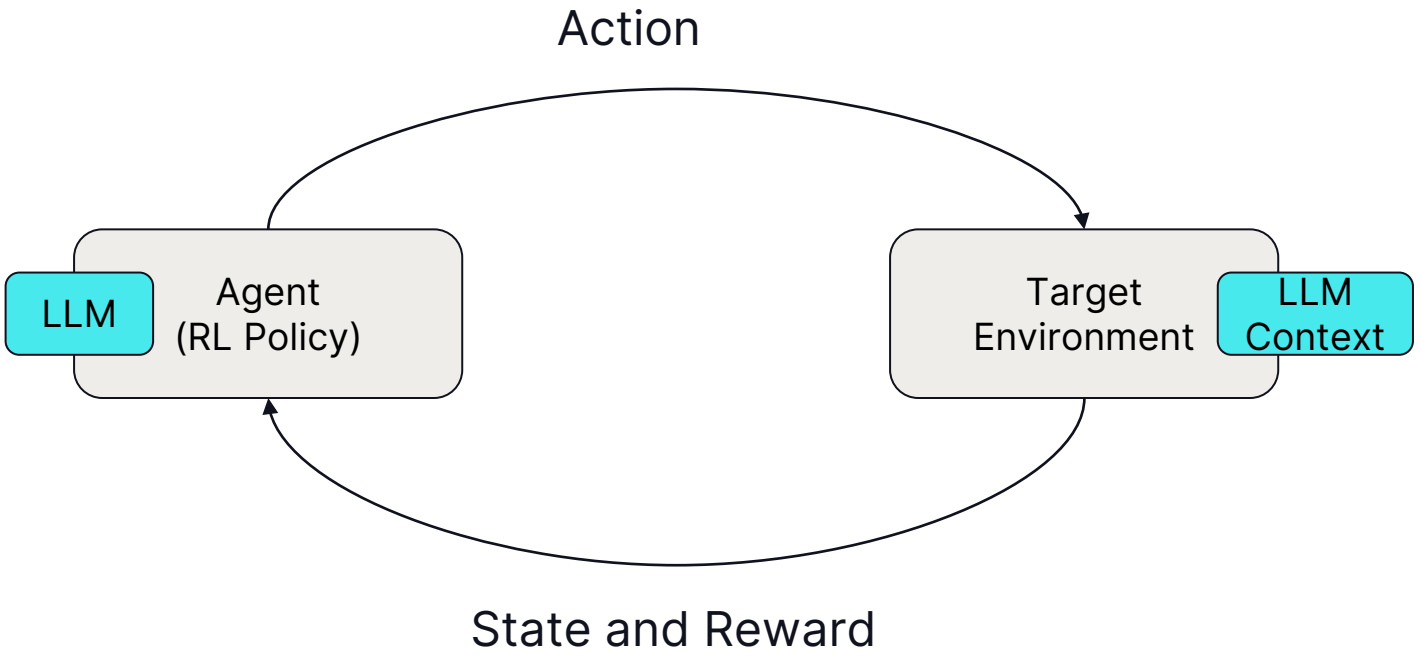
Using RLHF to align models

RL Recap



Using RLHF to align models

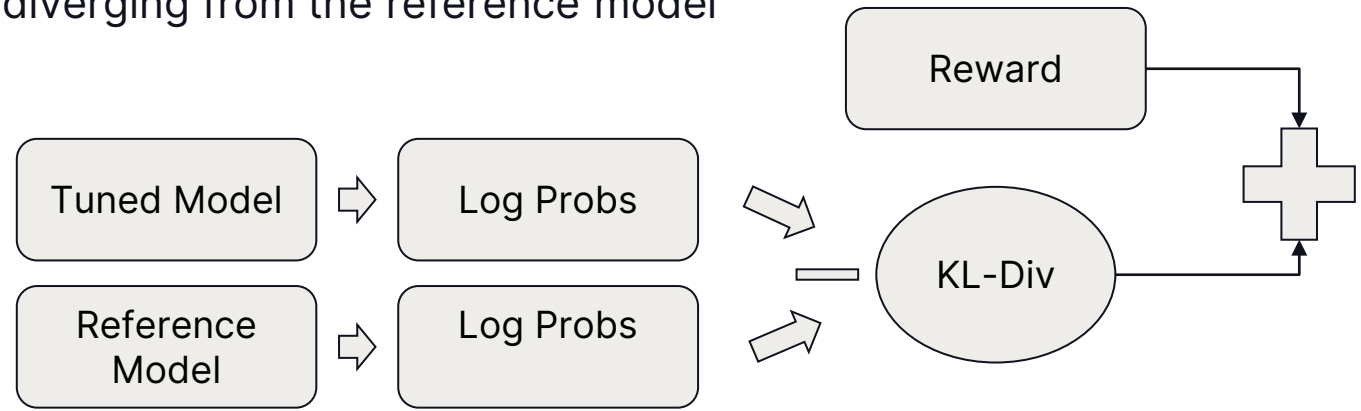
RL Recap



Using RLHF to align models

KL-Divergence

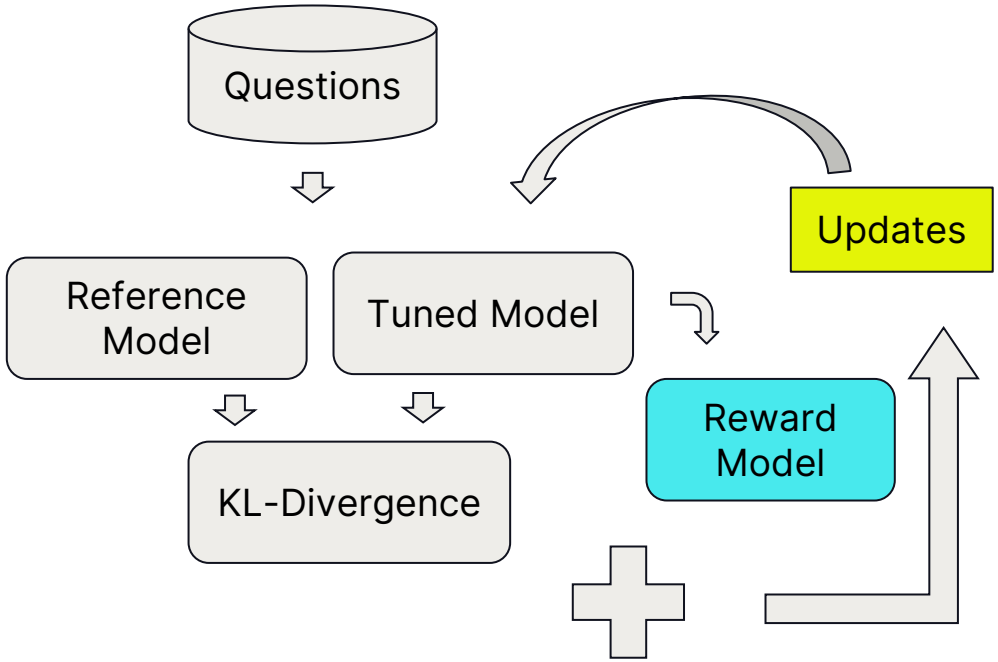
- To prevent “reward hacking” we do not want our model to diverge too much from the original model
- Kullback–Leibler(KL)-Divergence is a distance measure between distributions
- We add KL-Penalty term to the loss function to prevent our model from diverging from the reference model



Using RLHF to align models

Using PPO to fine-tune our model

- Policy gradients are applied to RL-Policy (Our tuned model)
- Some layers of the tuned models can be frozen
- We use PPO for the optimization:
  - On-policy RL algorithm
  - Supports both discrete and continuous action spaces
  - Scales well



Using RLHF to align models

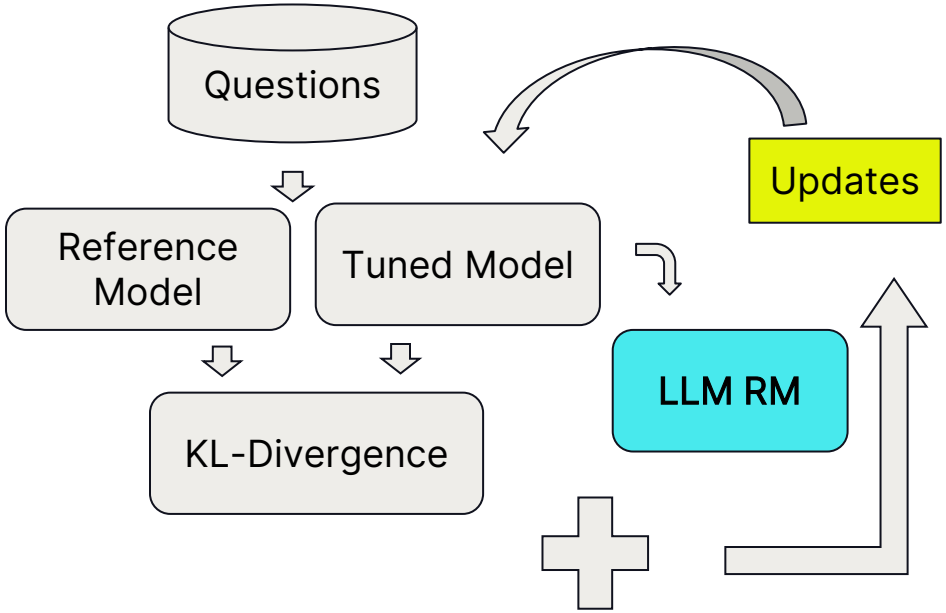
Gathering feedback data is hard and expensive

- It takes long time to gather high-quality feedback datasets
- If we are relying on feedback created by humans, we will need multiple versions (created by multiple people)
- Generating preference datasets can be not straightforward as well

RLAIF: Using LLM as a Reward Model

Let's use an LLM to provide feedback and replace the reward model with it

- We can use existing LLMs to provide feedback
- We could use this feedback either to train a Reward Model
- Or we can use an LLM as a Reward Model during the training directly



RLAIF is still very challenging

It’s still challenging to use PPO for RLAIF

- PPO is very resource intensive
- It’s hard to find the right hyperparameters combination to make PPO training stable
- PPO still requires a reward model

DPO:Direct Preference Optimization

- Direct Preference Optimization (DPO) aligns large language models with human preferences by directly optimizing the model's policy using **paired preference data**.
- **No need for the Reward Model:** DPO eliminates the need for a separate reward model and complex policy optimization by integrating human preferences directly into the training loop.
- DPO is much simpler to implement compared to PPO
- There are several other new alternatives to DPO which also rely on paired preference data: KTO, CPO, ORPO, etc

Practical Example

Model Alignment Solution Accelerator

Get started with Model Alignment really fast

- Project which you can use to get started with Model Alignment
- Implemented simple practical use case which can be easily adapted to many different business problems

- `git clone --single-branch --branch v2 https://github.com/databricks-solutions/dais-2024-llm4good.git`

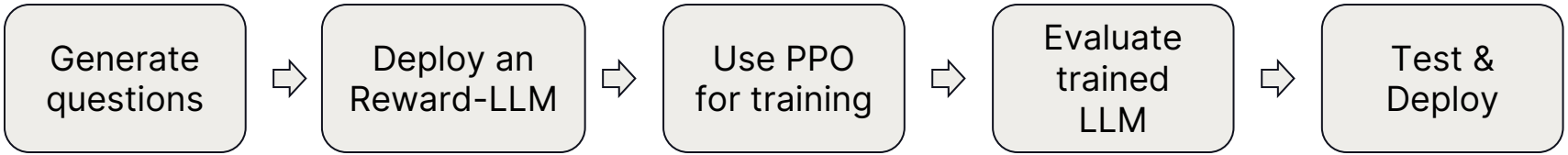
Use Case Description

Let’s try to apply these idea to a very simple business use case

- Let’s assume we are working for a grocery chain for vegetarians
- We want to build a chat bot which can help our customers with simple questions about our what they can cook with our items, etc
- But since our customers are vegetarians we do not want the chat bot to suggest any recipes or ingredients which contain meat or anything vegetarian-unfriendly

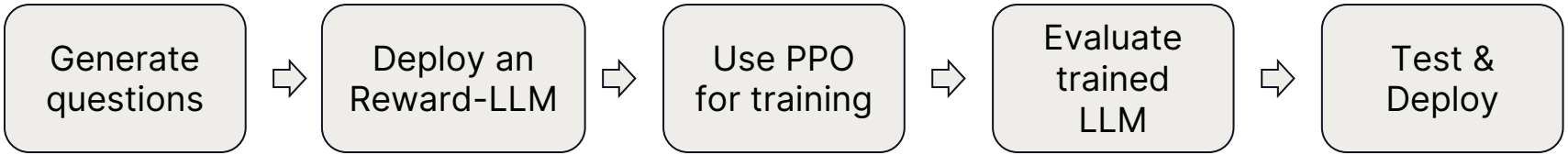
Let’s train vegetarian friendly LLM

- We will use PPO first
- We can use LLM-as-a-Judge approach to evaluate trained LLM

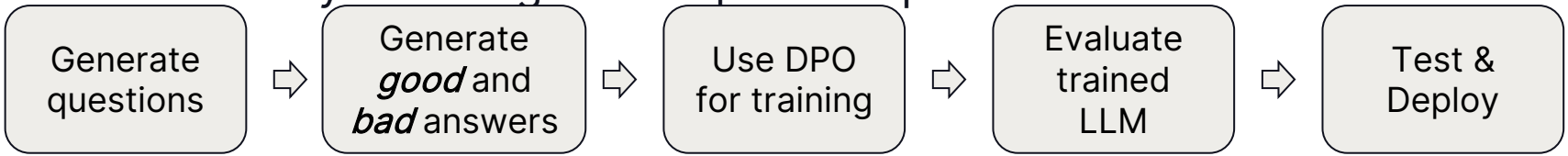


Let's train vegetarian friendly LLM

- We will use PPO first
- We can use LLM-as-a-Judge approach to evaluate trained LLM



- Alternatively we can generate pairwise preference data and use DPO



Implementation details

Modelling & Training

- TRL is a great open-source library which implements several alignments algorithms: https://github.com/huggingface/trl
- We will use:
  - PPO and DPO algorithms (implemented in TRL) for aligning Llama 3 8B model.
  - Llama 3 70B as a Reward Model
- We will also use Full Fine-tuning and LoRA with PPO

Implementation details

Data Generation (I)

- We will generate questions and answers using Llama 3 70B
- For batch question and answers generation we will can use:
    - Databricks Foundational Models API - Pay-per-token endpoints - works well for small datasets
    - Databricks Foundational Models API Provisioned Throughput Endpoint - the easiest option and scales well for medium datasets (5k-15k)
    - vLLM on Databricks or MCT cluster - might be preferable for the large scale generation

Implementation details

Data Generation (II)

- Even Llama 3 70B does not get a lot of questions from the first time
- To overcome this and make sure we get the correct answer:
  - We use the same LLM to calculate the score of the answer
  - If it’s not good, we ask the same LLM to improve it
- This makes sure we have high-quality data
- But it takes long time and is expensive
- Note: we do not need this step for PPO

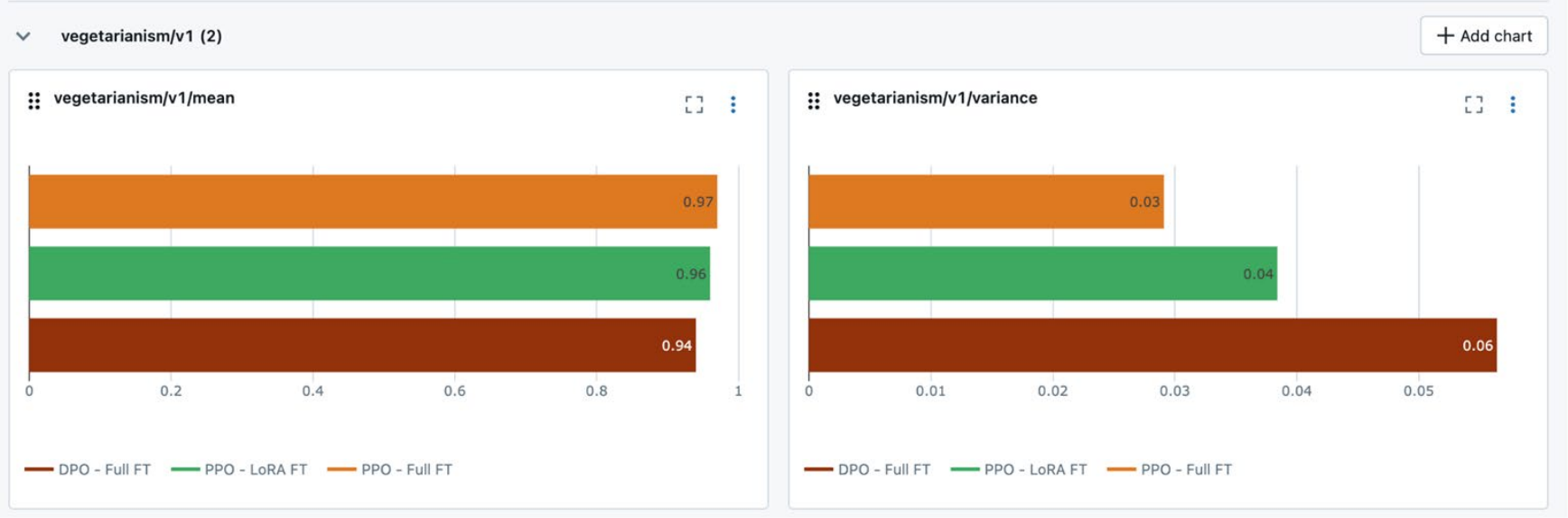
Implementation details

Model Evaluation

- For the final model evaluation we use LLM-as-a-Judge approach
- We will use LLama 3 70B as as Judge
- In a real scenario, it’s recommended to use even bigger/more powerful model (if this is possible)
- As a last step it’s recommended to use Human Experts to evaluate the model
- Agent Quality Framework is a great tool for that

Test results

Comparison of different approaches



Test results

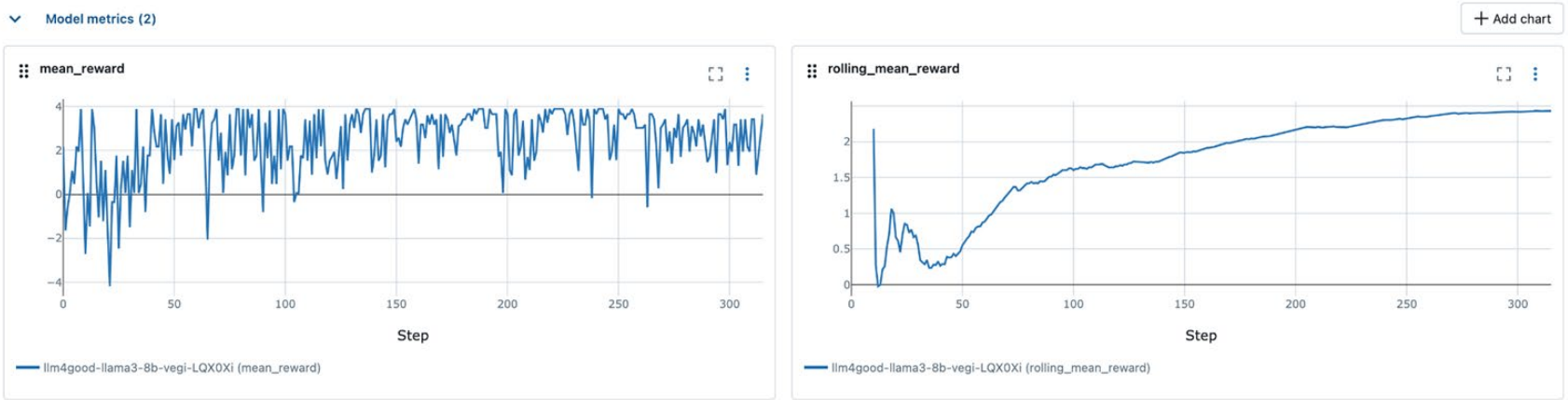
Comparison of different approaches

- It’s not a comparison of algorithms.
- The results do not mean that PPO is superior or DPO is worse even for this particular problem
- The questions and both good and bad answers were generated by the LLM, so with high quality human feedback DPO might outperform PPO
- In this particular scenario with generated data the benefits of DPO are not particular clear, since combined DPO training and data generation takes longer as PPO training

Important metrics

Comparison of different approaches

- It’s not a comparison of algorithms.



Model Alignment SA is an easy start

- You can clone and adjust Model Alignment Solution Accelerator and adjust it to your business requirements and align your LLM on Databricks or Mosaic MCT

- `git clone --single-branch --branch v2 https://github.com/databricks-solutions/dais-2024-llm4good.git`

DEMO

Thank you!
Questions?