dagster

Building Breakthrough AI Applications with Not Diamond

Not Diamond × Dagster

Feb 11 2025

Table of contents

dagster

Introductions

Meet the presenters

Tomás Hernando Kofman

Co-founder & CEO

Not Diamond

Alejandro Companioni

Founding Engineer

Not Diamond

Colton Padden

Developer Advocate

Dagster Labs

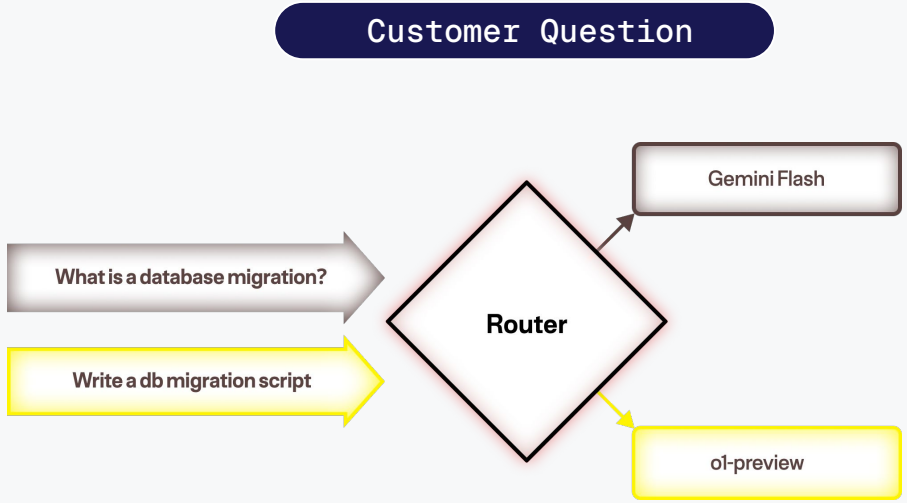
dagster

LLM Routing

LLM Routing

The best model on every input

As the number of models increases exponentially, selecting the right model for each request becomes increasingly challenging. LLM routing solves this problem by intelligently directing each query for you in real time, outperforming any individual model while reducing costs and latency by using smaller models when doing so doesn't degrade quality.



dagster

Why LLM routers are the future of prompting

`Intelligent Model Usage`

dagster

Always use the most effective model

Not every model is created equally. LLM routing ensures that your prompt is automatically directed to the model to produce the best result. This replaces suboptimal human heuristics with data-driven model selection.

Optimize for quality, cost, and speed

LLM performance varies significantly based on the type of request and costs can quickly escalate based on your model choice. Using an LLM router allows you to weigh your preferences between quality, cost, and speed.

AI Engineering & Data Engineering

AI engineering *is* data engineering

Data engineering tools lead to resilient AI pipelines

Working with LLMs and model inference fits nicely into the traditional tools of data pipeline building, allowing AI engineers to take advantage of a plethora of tools.

dagster

LLMs can empower data engineers

dagster

LLMs give data engineers the ability to make sense of unstructured organic data

It's easier than ever to make sense of messy and organic datasets. By introducing a step in your data processing pipelines that call out to AI services, engineers can add structure to data that was traditionally difficult to process.

How are data engineers using LLMs today?

Extracting insights

Finding trends and anomalies in data

Parsing unstructured data

Giving structure to non-deterministic organic data and classifying parsed data

Documentation and code generation

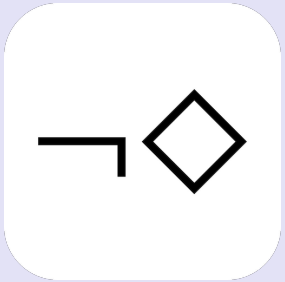
Metadata for data cataloging, like generated asset descriptions.

Troubleshooting

Debugging of failures during operation, or in development

dagster

Not Diamond



The backstory of Not Diamond

- Not Diamond was founded on the bet that the future of AI won't have a single, giant model. Instead, we'll have a decentralized ecosystem of highly specialized models. This will lead to higher-performing, more computationally efficient, and safer AI.
- As we pursued this bet, we increasingly saw teams at both Global 500s and leading startups struggling to figure out when to use which model.
- The more developers we talked to, the more conviction we built that multi-model infrastructure would be critical to moving off traditional AI workflows that waterfall development through one model at a time and towards data-driven, multi-path development.
- Our team is made of serial founders, AI researchers, and ML engineers with decades of experience in the field, and we're backed by leading AI engineers and scientists like Jeff Dean, Julien Chaumond, Ion Stoica, and Tom Preston-Werner.

Features

Model Router

Leverage intelligent, real-time routing that dynamically recommends the best model for each query to maximize quality, cost-efficiency, and latency, delivering optimal performance at scale.

Model Gateway

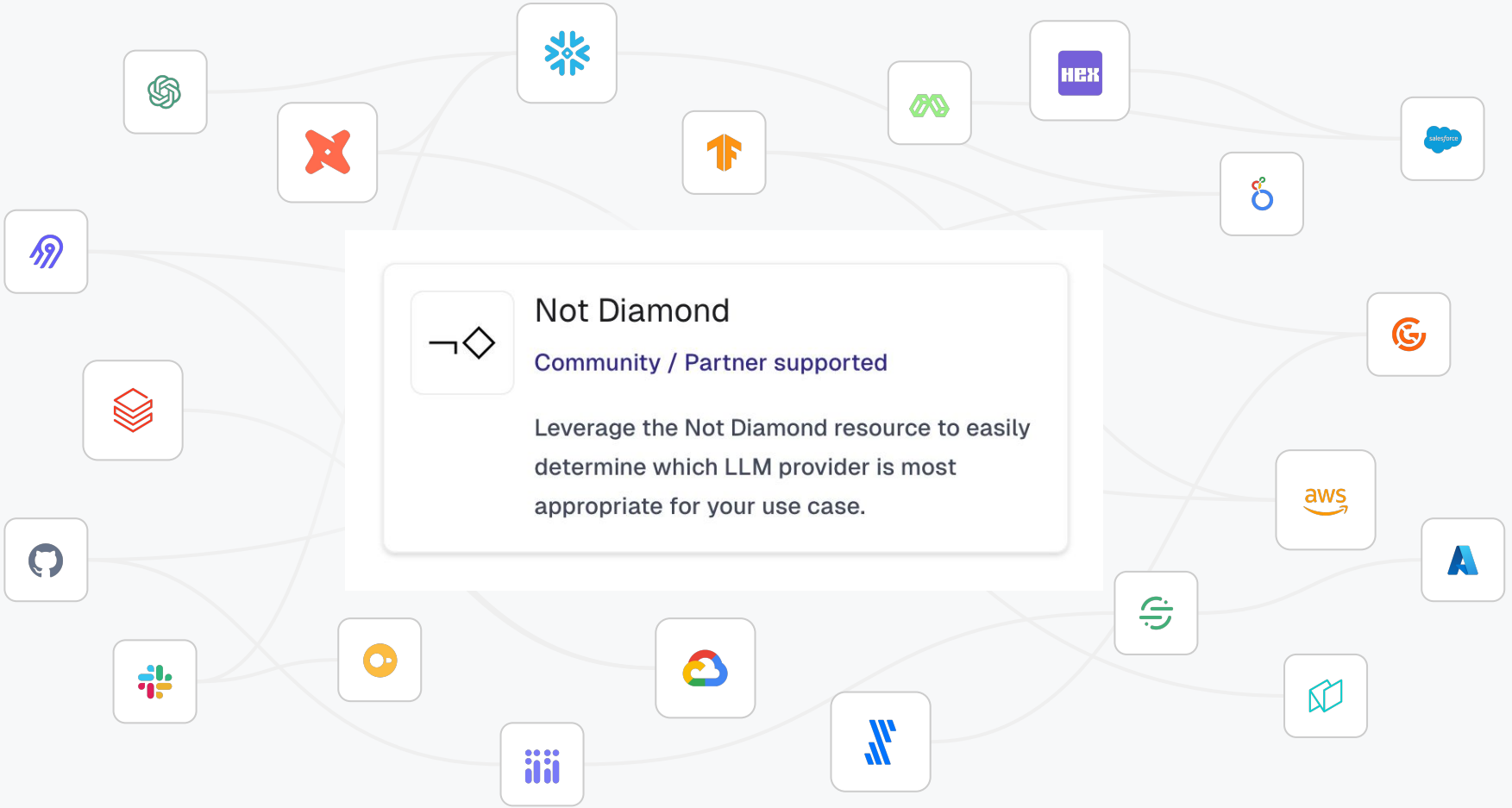
Instead of orchestrating every request to each LLM provider separately, you can use Not Diamond's gateway to seamlessly leverage all of the most popular models with a unified API interface.

Prompt Adaptation

Beta

Model selection and prompting are very tightly coupled. Not Diamond can help you optimize your prompt to each model to ensure you're using the best model with the best prompt on every input.

The `dagster-notdiamond` Integration


Not Diamond
Community / Partner supported
Leverage the Not Diamond resource to easily determine which LLM provider is most appropriate for your use case.
HEX
salesforce
aws

pip install dagster-notdiamond

The Not Diamond Resource

The `dagster-notdiamond` integration provides a resource that wraps the Not Diamond client.

This allows you to easily leverage the power of Not Diamond model routing from your assets, determining the best model for a prompt, and using that in tandem with integrations like `dagster-openai`, `dagster-anthropic`, or other providers.

However, Not Diamond can also be used directly from the OpenAI client through the *model gateway.*

dagster

```python
import dagster_notdiamond as nd


notdiamond_resource = nd.NotDiamondResource(
    api_key=dg.EnvVar("NOTDIAMOND_API_KEY")
)

defs = dg.Definitions(
    assets=[...],
    resources={
        "notdiamond": notdiamond_resource
    }
)
```

Integration usage: Model routing

Model routing

Both the `NotDiamondResource` and `OpenAIResource` are registered for use in our assets.

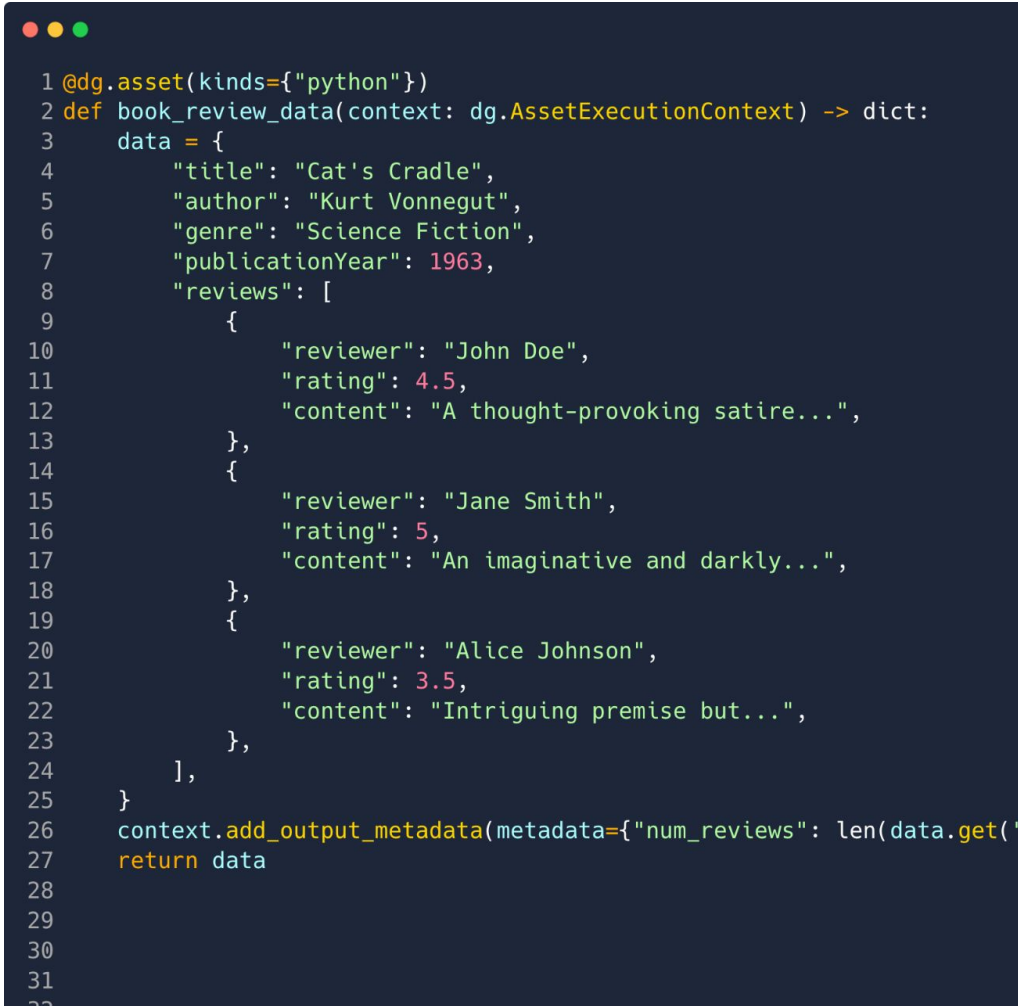
```python
import dagster as dg
import dagster_notdiamond as nd
import dagster_openai as oai


defs = dg.Definitions(
    assets=[book_review_data, book_reviews_summary],
    resources={
        "notdiamond": nd.NotDiamondResource(
            api_key=dg.EnvVar("NOTDIAMOND_API_KEY")
        ),
        "openai": oai.OpenAIResource(
            api_key=dg.EnvVar("OPENAI_API_KEY")
        ),
    },
)
```

dagster

Model routing

One approach to using Not Diamond is in tandem with integrations with other providers, for example `dagster-openai`.

Let's imagine we have a dataset of book reviews that we would like to summarize.



```python
@dg.asset(kinds={"python"})
def book_review_data(context: dg.AssetExecutionContext) -> dict:
    data = {
        "title": "Cat's Cradle",
        "author": "Kurt Vonnegut",
        "genre": "Science Fiction",
        "publicationYear": 1963,
        "reviews": [
            {
                "reviewer": "John Doe",
                "rating": 4.5,
                "content": "A thought-provoking satire...",
            },
            {
                "reviewer": "Jane Smith",
                "rating": 5,
                "content": "An imaginative and darkly...",
            },
            {
                "reviewer": "Alice Johnson",
                "rating": 3.5,
                "content": "Intriguing premise but...",
            },
        ],
    }
    context.add_output_metadata(metadata={"num_reviews": len(data.get('
    return data
```

dagster

Model routing

One approach to using Not Diamond is in tandem with integrations with other providers, for example `dagster-openai`.

Let's imagine we have a dataset of book reviews that we would like to summarize.

```python
@dg.asset(
    kinds={"openai", "notdiamond"}, automation_condition=dg.AutomationCondition.eager()
)
def book_reviews_summary(
    context: dg.AssetExecutionContext,
    notdiamond: nd.NotDiamondResource,
    openai: oai.OpenAIResource,
    book_review_data: dict,
) -> dg.MaterializeResult:

    prompt = f"""
    Given the book reviews for {book_review_data["title"]}, provide a detailed summary:

    {'|'.join([r['content'] for r in book_review_data["reviews"]])}
    """

    ...
```

dagster

Model routing

We pass our `prompt` to a method called `model_select` provided by the `notdiamond` resource.

This returns the best model to use when optimizing for *cost* as we specified `tradeoff="cost"`.

```python
@dg.asset(
    kinds={"openai", "notdiamond"}, automation_condition=dg.AutomationCondition.eag
)
def book_reviews_summary(
    context: dg.AssetExecutionContext,
    notdiamond: nd.NotDiamondResource,
    openai: oai.OpenAIResource,
    book_review_data: dict,
) -> dg.MaterializeResult:

    ...

    with notdiamond.get_client(context) as client:
        start = time.time()
        session_id, best_llm = client.model_select(
            model=["openai/gpt-4o", "openai/gpt-4o-mini"],
            tradeoff="cost",
            messages=[
                {"role": "system", "content": "You are an expert in literature"},
                {"role": "user", "content": prompt},
            ],
        )
        duration = time.time() - start

    ...
```

dagster

Model routing

Then, we take the `best_llm.model` returned by Not Diamond, and supply that to our call to OpenAI.

This allows us full control over the result of the model.

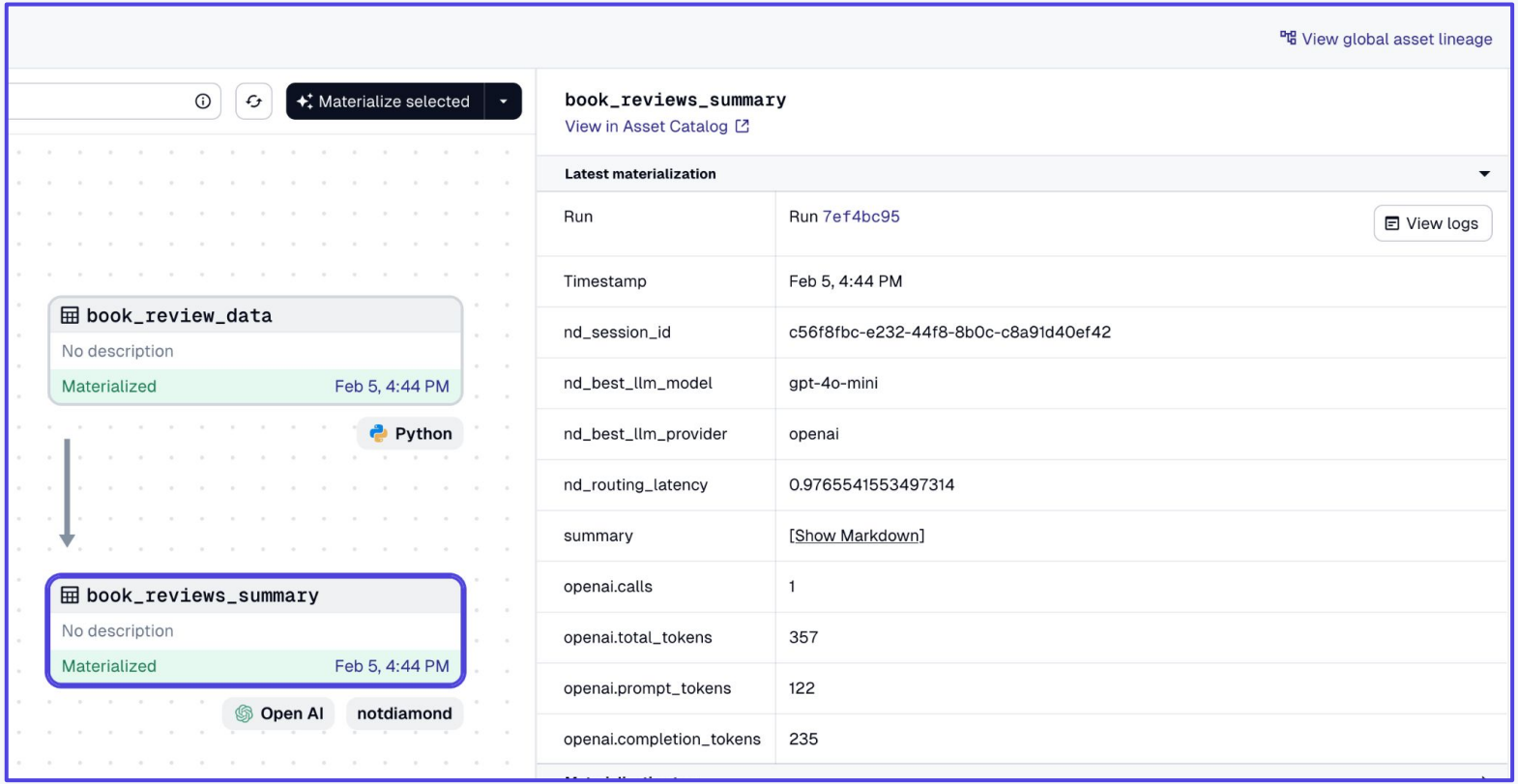
And in our materialized result, we include metadata around the LLM chosen.

dagster

```python
@dg.asset(
    kinds={"openai", "notdiamond"}, automation_condition=dg.AutomationCondition.eage
)
def book_reviews_summary(
    context: dg.AssetExecutionContext,
    notdiamond: nd.NotDiamondResource,
    openai: oai.OpenAIResource,
    book_review_data: dict,
) -> dg.MaterializeResult:

    ...

    with openai.get_client(context) as client:
        chat_completion = client.chat.completions.create(
            model=best_llm.model,
            messages=[{"role": "user", "content": prompt}],
        )

    summary = chat_completion.choices[0].message.content or ""

    return dg.MaterializeResult(
        metadata={
            "nd_session_id": session_id,
            "nd_best_llm_model": best_llm.model,
            "nd_best_llm_provider": best_llm.provider,
            "nd_routing_latency": duration,
            "summary": dg.MetadataValue.md(summary),
        }
    )
```

Model Routing

Example #1



dagster

Example #2

Integration usage: Model gateway

dagster

Model gateway

Only the `OpenAIResource` is required, but we configure the `api_key` and `base_url` to point to Not Diamond.

```python
import dagster as dg
import dagster_openai as oai


defs = dg.Definitions(
    assets=[book_review_data, book_reviews_summary],
    resources={
        "openai": oai.OpenAIResource(
            api_key=dg.EnvVar("NOTDIAMOND_API_KEY"),
            base_url="https://proxy.notdiamond.ai/v1/proxy",
        ),
    },
)
```

dagster

Model gateway

The model selection is handled for you!

You can provide `extra_body` parameters for additional options provided to Not Diamond.

```python
@dg.asset(
    kinds={"openai", "notdiamond"}, automation_condition=dg.AutomationCondi
)
def book_reviews_summary(
    context: dg.AssetExecutionContext,
    openai: oai.OpenAIResource,
    book_review_data: dict,
) -> dg.MaterializeResult:

    ...

    with openai.get_client(context) as client:
        chat_completion = client.chat.completions.create(
            model="notdiamond",
            extra_body={
              "models": ["gpt-4o", "claude-3-5-sonnet-20240620"],
              "tradeoff": "cost",
              "preference_id": "YOUR_PREFERENCE_ID"
            },
            messages=[{"role": "user", "content": prompt}],
        )

    ...
```

dagster

Discussion on LLM usage and best practices

Model routing has led to a 14% increase in our LLM output quality and a 72% reduction in our inference costs.

Tilen Babnik
CTO of Samwell

Summary

Summary

Not Diamond

Not Diamond provides a method for choosing the best LLM for your given use case.

The `dagster-notdiamond` integration makes it easy to leverage this service.

Model Routing

With the integration, you can provide your prompt, and Not Diamond will determine which model is most suitable, which you can then pass to your LLM provider of choice.

Model Proxying

Not Diamond also offers a proxying service, which automatically handles the routing, and providers requests for you behind the scenes.

References & Resources

01 **Not Diamond quickstart**

docs.notdiamond.ai/docs/quickstart

02 **Integration reference**

docs.dagster.io/integrations/libraries/notdiamond

03 **Integration source code**

github.com/dagster-io/community-integrations/.../dagster-notdiamond

dagster

Q & A

Subtitle goes here

Thank you!

The advantage of data-driven LLM usage

`Intelligent Model Usage`

Optimize routing with your data

Like everything in AI, data is the driving force of every outcome. Create a custom router by providing your dataset to tune the various paths each request can take. Let the router do the hard work with some subtle guidance.

Real-time tuning for model routing

Part of being human is having preferences and current models don't do a great job of personalizing the outcome. Let your users' preferences influence the model selection by reacting to real-time feedback.

dagster