

Beam ML past, present and future

Kerry Donny-Clark & Reza Rokni



I'm Kerry





I'm Reza





Agenda



- Intro
- The Past: RunInference is born
- The Present: Model handlers, model updates, model zoos, and more!
- The Future: Make ML tasks easy
- Q&A

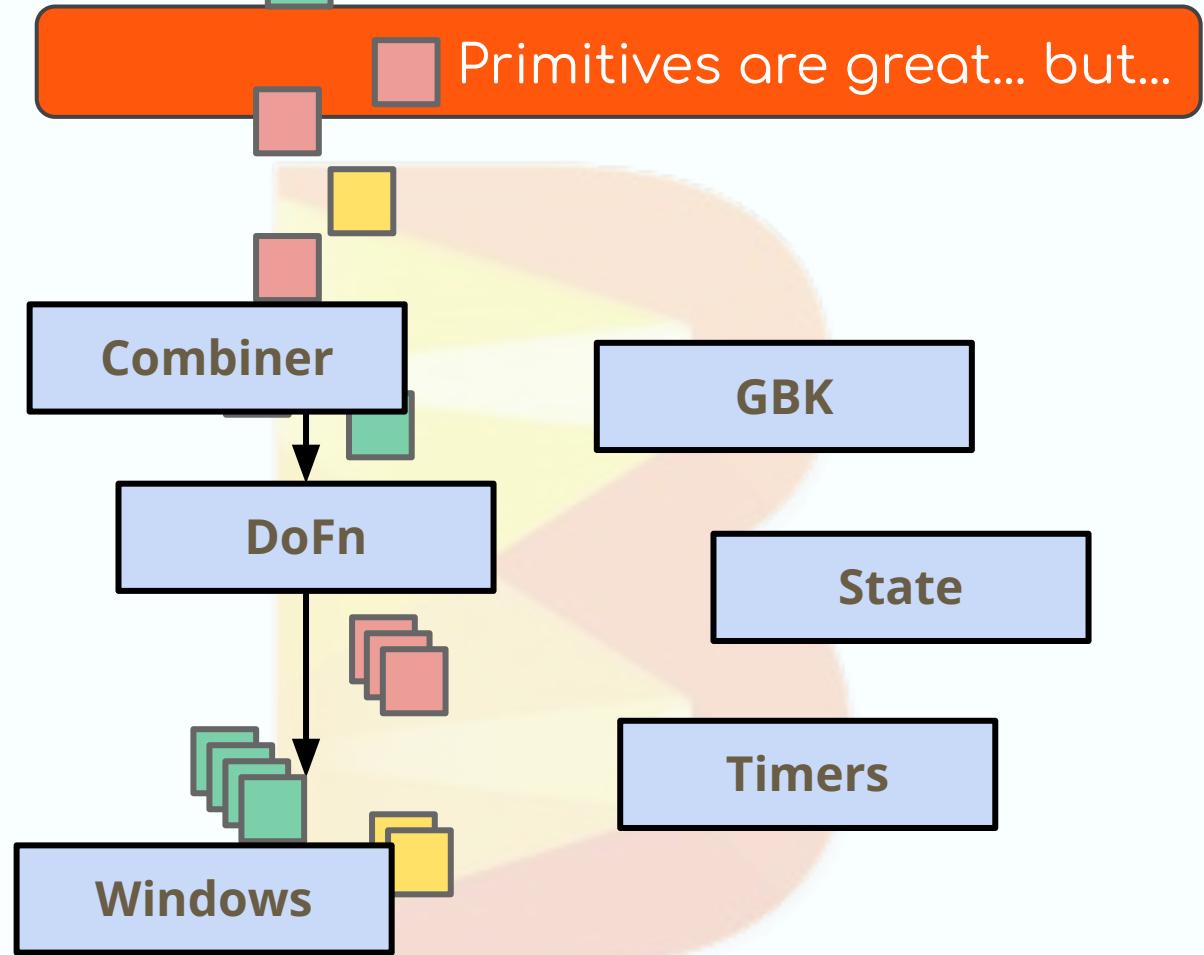


Turn key solutions....



Apache Beam

Java	<pre>Input.apply (Sum.integersPerKey())</pre>
Python	<pre>input Sum.PerKey()</pre>
SQL	<pre>SELECT key, SUM(value) FROM input GROUP BY key</pre>
Go	<pre>stats.Sum(s, input)</pre>



Apache Beam

Primitives are great... but...



Patterns are great ... but

The screenshot shows a portion of the Beam website's documentation page. At the top, there is a navigation bar with links for "About", "Get Started", "Documentation", "Roadmap", and "Community". Below the navigation bar is a blue header bar with the "BEAM SUMMIT" logo and a cartoon character icon. The main content area has a white background and features the text "DOCUMENTATION" and "Common pipeline patterns". Below "Common pipeline patterns" is a link "Using the Documentation".

beam About Get Started Documentation Roadmap Community

BEAM SUMMIT

DOCUMENTATION

Common pipeline patterns

Using the Documentation

Patterns are great ... but

The screenshot shows a portion of the Beam Summit website. At the top, there is a navigation bar with links for "About", "Get Started", "Documentation", "Roadmap", and "Community". Below the navigation bar is a blue header section with a yellow dotted line on the left and the "BEAM SUMMIT" logo on the right, which features a cartoon character holding a tablet displaying a chart. In the center of this blue section, the text "Common pipeline patterns" is displayed. To the left of this central text, the word "DOCUMENTATION" is visible above a link labeled "Using the Documentation".



Which set of patterns do I need...?

Patterns are great ... but

The screenshot shows a portion of the Beam website. At the top, there is a navigation bar with links for "About", "Get Started", "Documentation", "Roadmap", and "Community". Below the navigation bar is a blue header section with a yellow dotted line on the left. In the center of this section is the "BEAM SUMMIT" logo, which consists of the word "BEAM" in large orange letters above the word "SUMMIT" in smaller white letters. To the right of the logo is a cartoon illustration of a red robot or character standing next to a small screen displaying a bar chart. The main content area below the header has a white background. On the left side of this area, the word "DOCUMENTATION" is written in capital letters. In the center, the text "Common pipeline patterns" is displayed in a large, bold, dark font. At the bottom left of the content area, there is a smaller text link that says "Using the Documentation".



Set reminder..

Patterns are great ... but

The screenshot shows a website header with navigation links: beam (with a logo), About, Get Started, Documentation (with a dropdown arrow), Roadmap, and Community. Below the header is a blue banner with a yellow dotted line on the left, the BEAM SUMMIT logo in the center, and a cartoon character on the right. The main content area has a white background. On the left, there's a 'DOCUMENTATION' section with a link 'Using the Documentation'. On the right, the title 'Common pipeline patterns' is displayed in large, bold, dark font.



Writing that code!



Writing same code!



Yup same again!



Déjà vu?

Turn Key transforms

```
with beam.Pipeline(options=pipeline_options) as p:  
    (p  
        | beam.io.fileio.MatchFiles(gs://my_bucket/*)  
        | beam.io.fileio.ReadMatches()  
        | beam.Map(preprocess_image)  
        | beam.xxx_pattern_xxx(Configuration)  
        ...)
```

credit karma®

Self-service Machine Learning Workflows and Scaling MLOps with Apache Beam

Apache Beam has future-proofed Credit Karma's data and ML platform for scalability and efficiency, enabling MLOps with unified pipelines, processing 5-10 TB daily at 5K events per second, and managing 20K+ ML features.



Real-time ML with Beam at Lyft

Lyft Marketplace team aims to improve our business efficiency by being nimble to real-world dynamics. Apache Beam has enabled us to meet the goal of having a robust and scalable ML infrastructure for improving model accuracy with features in real-time. These real-time features support critical functions like

Large-Scale Generation of ML Podcast Previews at Spotify with Google Dataflow



April 13, 2023

Published by Diego Casabuena (ML Engineer, Spotify), Edgar Tanaka (ML Engineer, Spotify), Winstead Zhu (ML Engineer, Spotify), Reza Rokni (PM, Google Dataflow ML), and Danny McCormick (Senior Software Engineer, Google)



Integrating the Podz ML pipeline into Spotify

Lots of community engagement!

Beam Summit 2022:

Machine learning design patterns between Beam and a hard place	Lak Lakshmanan		
Vega: Scaling MLOps Pipelines at Credit Karma using Apache Beam and Dataflow	Debasish Das, Vishnu Venkataraman & Raj Katakam		
Implementing Cloud Agnostic Machine Learning Workflows with Apache Beam on Kubernetes	Charles Adetiloye & Alexander Lerma		
Streaming NLP infrastructure on Dataflow	Alex Chan & Angus Neilson		
Improving Beam-Dataflow Pipelines for Text Data Processing	Sayak Paul & Nilabhra Roy Chowdhury		
The Ray Beam Runner Project: A Vision for Unified Batch, Streaming, and ML	Patrick Ames, Jiajun Yao & Chandan Prasad		

Turn Key Transformations

Beam RunInference

Removing boiler plate chore!

```
with beam.Pipeline(options=pipeline_options) as p:  
    (p  
        | beam.io.fileio.MatchFiles(gs://my_bucket/images*)  
        | beam.io.fileio.ReadMatches())
```

Removing boiler plate chore!

```
with beam.Pipeline(options=pipeline_options) as p:  
    (p  
        | beam.io.fileio.MatchFiles(gs://my_bucket/images*)  
        | beam.io.fileio.ReadMatches()  
        | beam.Map(preprocess_image)
```

Removing boiler plate chore!

```
class MyComplicatedPredictionStuff(beam.DoFn):
```

Removing boiler plate chore!

```
class MyComplicatedPredictionStuff(beam.DoFn):  
    def setup():  
        #Code for loading once  
        ...  
  
    def process(self, element):  
        #Use model handle to call
```

Removing boiler plate chore!

```
class MyComplicatedPridctionStuff(beam.DoFn):  
    def setup():  
        #Code for loading once  
        ...  
  
    def process(self, element):  
        #Use model handle to call  
        ...  
        #Handle errors, do nice error logging
```

Removing boiler plate chore!

```
class MyComplicatedPredictionStuff(beam.DoFn):  
    def setup():  
        #Code for loading once  
        ...  
  
    def process(self, element):  
        #Use model handle to call  
        ...  
        #Handle errors, do nice error logging  
        ...  
        #Output useful metrics from the process  
        ...
```

Removing boiler plate chore!

```
class MyComplicatedPredictionStuff(beam.DoFn):  
    def setup():  
        #Code for loading once  
        ...  
  
    def process(self, element):  
        #Use model handle to call  
        ...  
        #Handle errors, do nice error logging  
        ...  
        #Output useful metrics from the process  
        ...  
        TODO Oh wait! I need to batch stuff first ...
```

Removing boiler plate chore!

```
class MyComplicatedPridctionStuff(beam.DoFn):  
    def setup():  
        TODO Code for loading once .....,  
  
    def process(self, element):  
        TODO Use model handle to call  
        TODO Handle errors, do nice error logging  
        TODO Output useful metrics from the process  
        TODO Oh wait! I need to batch stuff first ...  
TODO Wait.. I need model configuration ....
```

Removing boiler plate chore!

```
with beam.Pipeline(options=pipeline_options) as p:  
    (p  
        | beam.io.fileio.MatchFiles(gs://my_bucket/images*)  
        | beam.io.fileio.ReadMatches()  
        | beam.Map(preprocess_image)  
        | beam.ml.inference.RunInference(model_handler)  
        ...
```

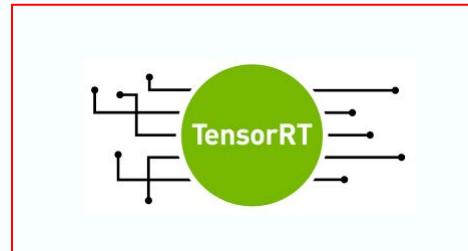
The Present: Model Handlers

PytorchModelHandlerTensor

Framework

Type

The Present: Model Handlers



The Present: Notebooks

<https://github.com/apache/beam/tree/master/examples/notebooks/beam-ml>

- Prediction and inference with pretrained models
- Custom inference
- Automatic Model Refresh
- Multi-model pipelines
- Model Evaluation
- Data processing

DEMO

DEMO

[LLM Demo Link](#)

The Present: Model Zoos

CLASSIFIER_URL=

"<https://tfhub.dev/google/wiki40b-lm-en/1>"

with pipeline as p:

```
predictions = (  
    p
```

```
        | ReadFromText(known_args.input)  
        | RunInference(  
            TensorHubhandler(  
                {uri=CLASSIFIER_URL}  
            ))
```

TensorFlow Hub is a repository of trained machine learning models.

TensorFlow Hub is a repository of trained machine learning models ready for fine-tuning and deployable anywhere. Reuse trained models like BERT and Faster R-CNN with just a few lines of code.

See the guide
Learn about how to use TensorFlow Hub and how it works.

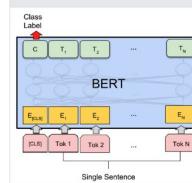
See tutorials
Tutorials show you end-to-end examples using TensorFlow Hub.

See models
Find trained TF, TFLite, and TF.js models for your use case.

```
!pip install --upgrade tensorflow_hub  
  
import tensorflow_hub as hub  
  
model = hub.KerasLayer("https://tfhub.dev/google/nnlm-en-dim128/2")  
embeddings = model([["The rain in Spain.", "falls",  
                    "mainly", "In the plain."])  
  
print(embeddings.shape) # (4, 128)
```

Models

Find trained models from the TensorFlow community on [TFHub.dev](#)



BERT

Check out BERT for NLP tasks including text classification and question answering.

See the model ↗



Object detection

Use the Faster R-CNN Inception ResNet V2 640x640 model for detecting objects in images.

See the model ↗



Style transfer

Transfer the style of one image to another using the image style transfer model.

See the model ↗



On-device food classifier

Use this TFLite model to classify photos of food on a mobile device.

See the model ↗

DEMO

DEMO

[notebooks/beam-ml/run_inference_with_tensorflow_hub.ipynb](#)

The Present: Model map() encapsulation

```
inference = pcoll | RunInference(model_handler.with_postprocess_fn(lambda x : do_something_to_result(x)))
```

```
inference = pcoll | RunInference(model_handler.with_preprocess_fn(lambda x : do_something(x)))
```

```
inference = pcoll | RunInference(  
    model_handler.with_preprocess_fn(  
        lambda x : do_something(x))  
    .with_preprocess_fn(  
        lambda x : do_something_else(x))  
    .with_postprocess_fn(  
        lambda x : do_something_after_inference(x))  
    .with_postprocess_fn(  
        lambda x : do_something_else_after_inference(x))  
)
```

The Present: Error handling

```
main, other = pcoll | RunInference(model_handler).with_exception_handling()  
other.failed_inferences | beam.Map(print) # insert logic to handle failed records here
```

```
main, other = pcoll | RunInference(model_handler.with_preprocess_fn(f1).with_postproces  
s_fn(f2)).with_exception_handling()  
other.failed_preprocessing[0] | beam.Map(print) # handles failed preprocess operations,  
indexed in the order in which they were applied  
other.failed_inferences | beam.Map(print) # handles failed inferences  
other.failed_postprocessing[0] | beam.Map(print) # handles failed postprocess operation  
s, indexed in the order in which they were applied
```

The Present: Streaming Model Updates

RunInference auto-model update

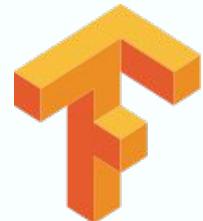
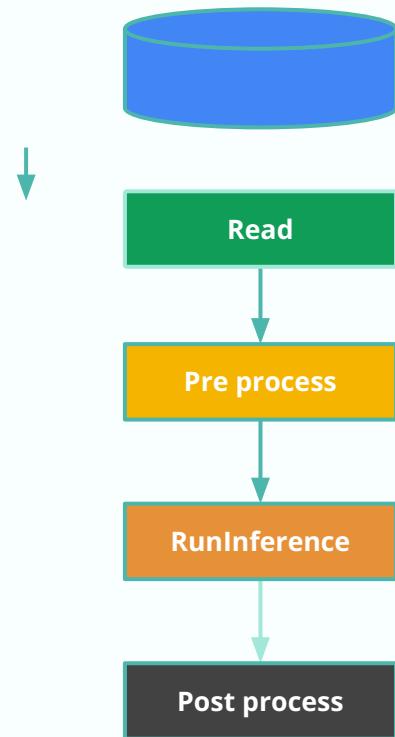
Two modes:

1. Watch Mode

Upload updated model to files stores like GCS and
RunInference will auto pull the new model for you

2. Event Mode

Push an update message to RunInference via a
streaming source such as Kafka

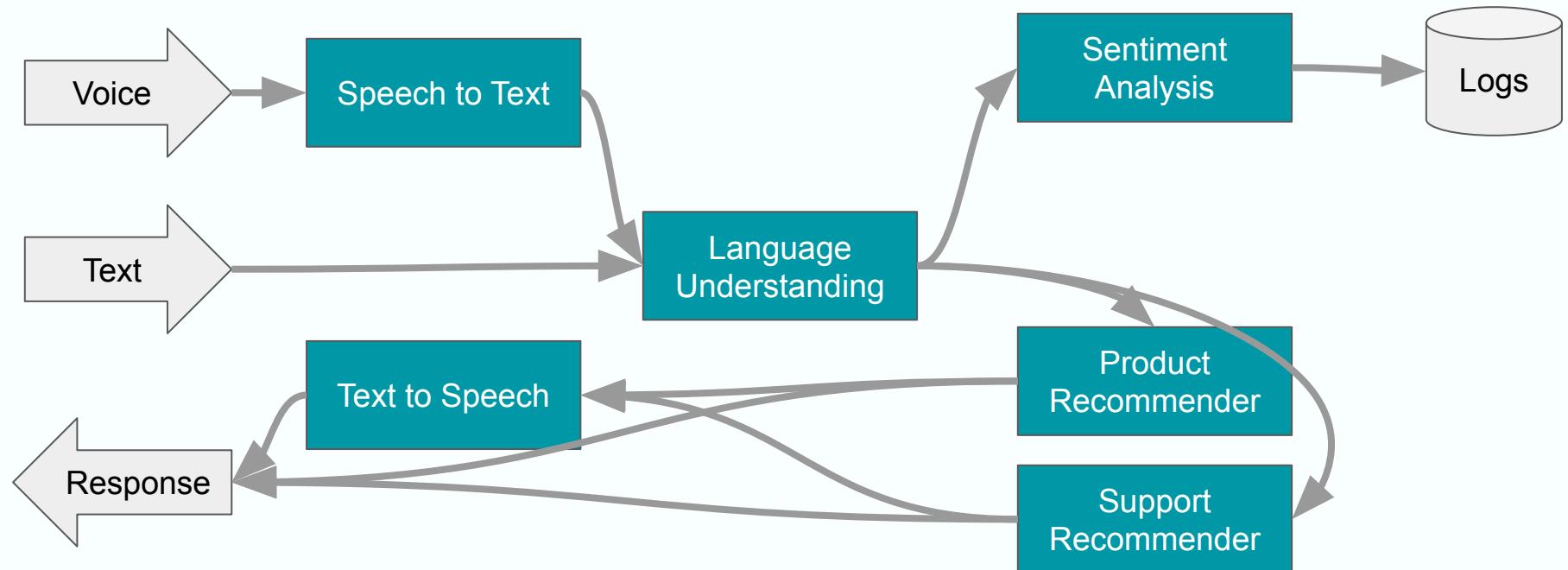


The Present: Efficient large models

share_model_across_processes() → bool [\[source\]](#)

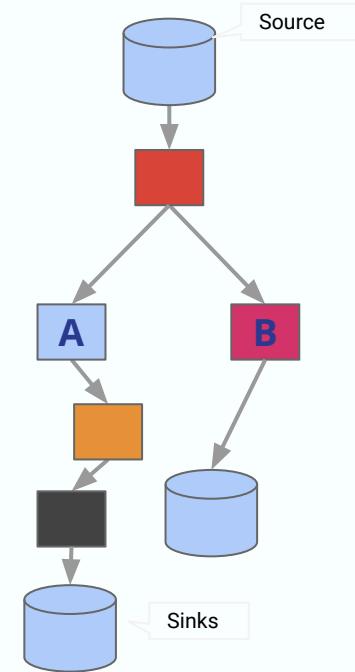
apache_beam.utils.multi_process_shared module

The Present: Multi-Model Ensembles



The Present: Branched (A/B) models

```
data = p | beam.io.textio(files)  
data | RunInference (model_a_handler)  
data | RunInference (model_b_handler)
```

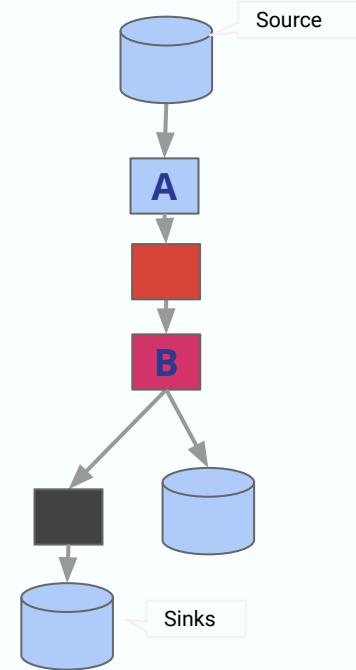
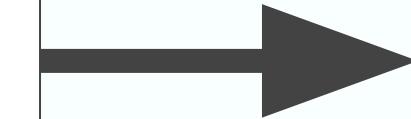


The Present: Sequential Models

```
data = p | beam.io.textio(files)

model_a_output =
    data | RunInference(model_a_handler)

model_a_output
| Map(postprocess)
| RunInference(model_b_handler)
```



The Future: More Integrations

Hugging Face Model Handler for RunInference

Ritesh Ghorse (riteshghorse@apache.org)

```
with pipeline as p:  
    predictions = (p  
        | beam.ReadFromSource('a_source')  
        | RunInference(  
            HuggingFaceModelHandler(...)))
```

Future : Models from endpoints

[WIP] Vertex AI Remote Model Handler #27091

 Draft

jrmccluskey wants to merge 3 commits into [apache:master](#) from [jrmccluskey:vertexAI](#) 

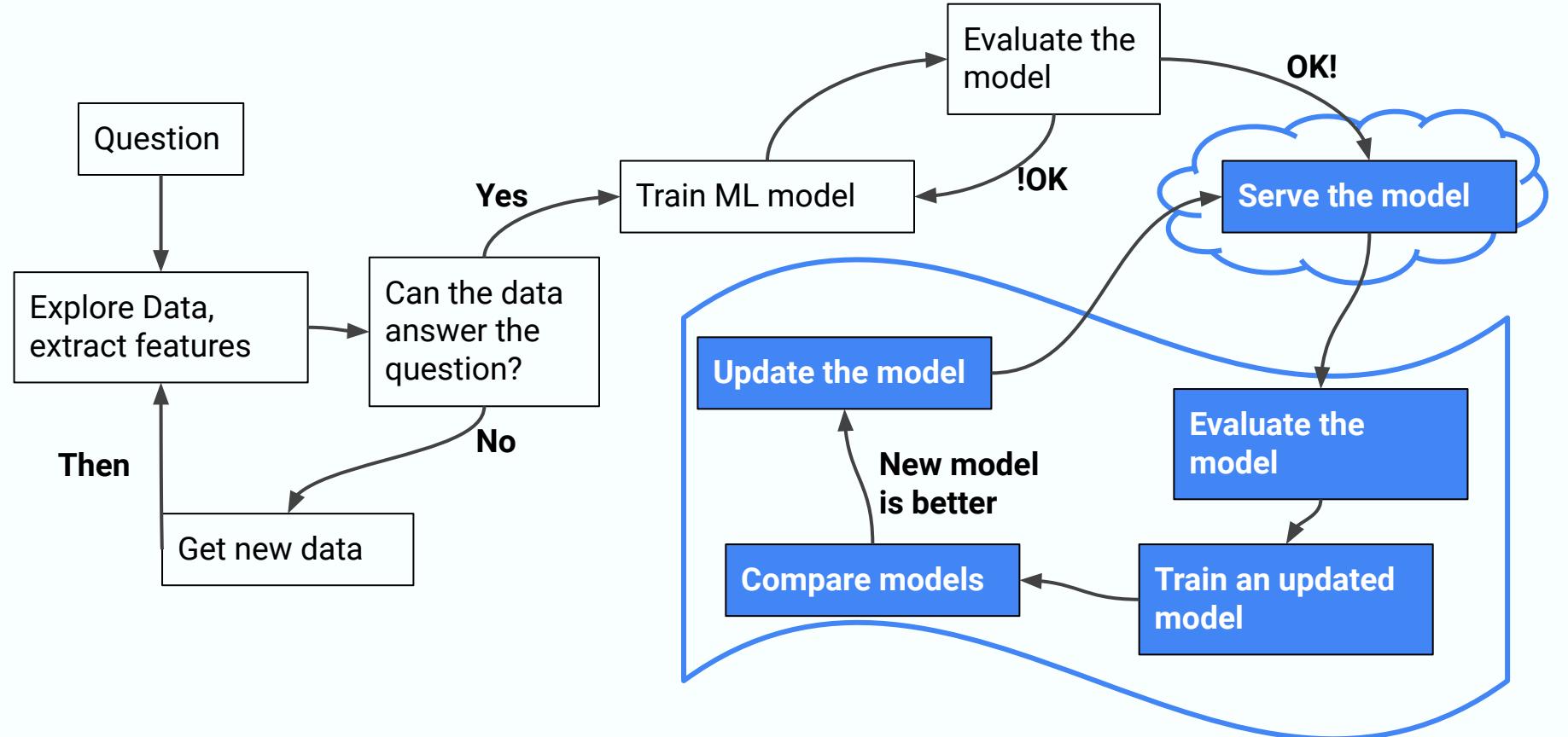


Beyond Inference !



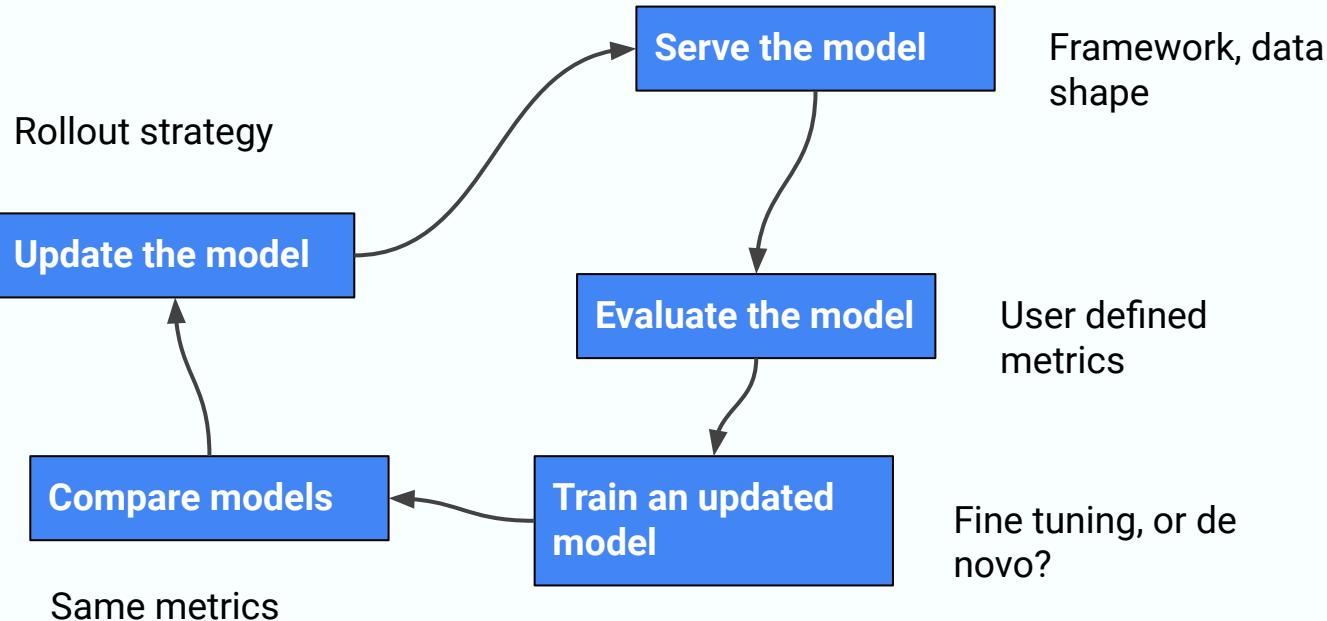


Beyond Inference !





Beyond Inference !



Coming soon

Beam MLTransform

<https://s.apache.org/beam-mltransform>

Anand Inguva (anandinguva@google.com)

Last updated: May 30th, 2023

WIP!

```
beam.MLTransform(  
    process_handler=ProcessHandler([  
        scale_to_0_1(  
            columns={'x': int, 'y': List[int]}),  
        compute_and_apply_vocab(  
            columns={'x': int, 'y': List[int]}))])
```

Turn Key Transformations

Coming soon



Reza Rokni & Kerry Donny-Clark

QUESTIONS?

B≡ΔM
S U M M I T