Auto model refresh in RunInference
Agenda

- RunInference
- Automatic model refresh in RunInference
RunInference

- RunInference is a simple to use PTransform that can be used for the task of ML Inference.
- RunInference helps users to avoid writing boilerplate code with the help of ModelHandler
  - ModelHandler - Framework specific modules which are required to configure parameters for the model.
  - Pytorch, Tensorflow, Sklearn, XGboost, Onnx and TensorRT are supported. Many more to come.
RunInference streamlines model loading, batching, and error handling for invalid inputs, while also calculating metrics like model loading latency and inference latency, and managing model sharing across threads in a process.

```python
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)
```
How to update RunInference ML model in a running beam pipeline?
Current process to update models

- RunInference pipelines uses a specific models for predictions
- Updating the model requires stopping the pipeline, changing the model path, and restarting the pipeline.
Issues with current process

- The pipeline interruption can lead to service downtime.
- The model update process is manual, leading to potential human error.
Introduce automatic model refresh

- Enables model updating without stopping the pipeline.
- The feature is automated, reducing the chance of errors.
How Automatic Model Refresh works?

- Uses Beam’s side inputs to fetch the latest model path.
- RunInference accepts a side input which should be a Singleton.

Side inputs

- Accessible from a DoFn
  - Elements that can determined during runtime.
- Side inputs can be used as a caching layer.
  - Store the model metadata such as model path, model id etc.

Reference to side inputs:

https://beam.apache.org/documentation/programming-guide/#side-inputs
Side inputs in RunInference
ModelMetaData

- RunInference expects that the side input passed has
  - PColl elements are wrapped around `ModelMetaData`
  - PColl view is Singleton

```python
class ModelMetadata(NamedTuple):
    model_id: str
    model_name: str
```

- `model_id`: URI or path to the ML model.
- `model_name`: a prefix to the metrics namespace to differentiate between the models.
Modes - Automatic model refresh

Watch mode

- You watch a directory for model updates.
- You can use Apache Beam provided patterns such as `WatchFilePattern`.

Event mode

- Use Pub/Sub to send model updates to the RunInference.
WatchFilePattern - A pattern of watch mode

- Watches a directory for a matching file_pattern.
- Specify interval in seconds to check for the matching file_pattern.
- Follows slowly updating side input pattern.
- Newly updated matching file name should be unique.

```python
with beam.Pipeline(options=pipeline_options) as p:
    watch_file_pattern = (p | WatchFilePattern(file_pattern=<your_glob_pattern>))
                        (p
                         | beam.io.fileio.MatchFiles(gs://my_bucket/images*)
                         | beam.io.fileio.ReadMatches()
                         | beam.Map(preprocess_image)
                         | beam.ml.inference.RunInference(
                             model_handler, model_metadata_pcoll=watch_file_pattern)
```
ReadFromPubSub - A pattern of Event mode

- Example: Use `ReadFromPubSub` to get the latest model path.
- Make sure the side input PCollection has a Singleton View.

```python
with beam.Pipeline() as p:
    event_model_side_input = (p
        | "ReadFromPubSub" >> ReadFromPubSub(topic='<your_topic>')
        | "ConvertToModelMetaData" >> beam.Map(
            lambda x: ModelMetaData(model_id=x,
                                    model_name=get_unique_name(model_name))
        )
    (p
        | beam.io.fileio.MatchFiles(gs://my_bucket/images*)
        | beam.io.fileio.ReadMatches()
        | beam.Map(preprocess_image)
        | beam.ml.inference.RunInference(
            model_handler,
            model_metadata_pcoll= event_model_side_input)
```
Output of RunInference

- **PredictionResult** - A NamedTuple
  - example
  - inference
  - `model_id`: used to differentiate between different models.
Summary

- Use automatic model refresh to update your models in a streaming pipeline.
- WatchMode and EventMode
- Use beam provided patterns such as WatchFilePattern.
QUESTIONS?

Gmail: anandinguva98@gmail.com, anandinguva@google.com.
Linkedin:
https://www.linkedin.com/in/anandinguva-5689a1128/