#include "tensorflow/core/framework/op.h"

#include "tensorflow/core/framework/shape\_inference.h"

#include "tensorflow/core/framework/op\_kernel.h"

#include <twml.h>

#include "tensorflow\_utils.h"

using namespace tensorflow;

REGISTER\_OP("BatchPredictionResponseWriter")

.Attr("T: {float, double}")

.Input("keys: int64")

.Input("values: T")

.Output("result: uint8")

.SetShapeFn([](::tensorflow::shape\_inference::InferenceContext\* c) {

return Status::OK();

}).Doc(R"doc(

A tensorflow OP that packages keys and values into a BatchPredictionResponse.

values: input feature value. (float/double)

keys: feature ids from the original BatchPredictionRequest. (int64)

Outputs

bytes: output BatchPredictionRequest serialized using Thrift into a uint8 tensor.

)doc");

template<typename T>

class BatchPredictionResponseWriter : public OpKernel {

public:

explicit BatchPredictionResponseWriter(OpKernelConstruction\* context)

: OpKernel(context) {}

void Compute(OpKernelContext\* context) override {

const Tensor& keys = context->input(0);

const Tensor& values = context->input(1);

try {

// Ensure the inner dimension matches.

if (values.dim\_size(values.dims() - 1) != keys.dim\_size(keys.dims() - 1)) {

throw std::runtime\_error("The sizes of keys and values need to match");

}

// set inputs as twml::Tensor

const twml::Tensor in\_keys\_ = TFTensor\_to\_twml\_tensor(keys);

const twml::Tensor in\_values\_ = TFTensor\_to\_twml\_tensor(values);

// no tensors in this op

const twml::Tensor dummy\_dense\_keys\_;

const std::vector<twml::RawTensor> dummy\_dense\_values\_;

// call constructor BatchPredictionResponse

twml::BatchPredictionResponse tempResult(

in\_keys\_, in\_values\_, dummy\_dense\_keys\_, dummy\_dense\_values\_);

// determine the length of the result

int len = tempResult.encodedSize();

TensorShape result\_shape = {1, len};

// Create an output tensor, the size is determined by the content of input.

Tensor\* result = nullptr;

OP\_REQUIRES\_OK(context, context->allocate\_output(0, result\_shape,

&result));

twml::Tensor out\_result = TFTensor\_to\_twml\_tensor(\*result);

// Call writer of BatchPredictionResponse

tempResult.write(out\_result);

} catch(const std::exception &e) {

context->CtxFailureWithWarning(errors::InvalidArgument(e.what()));

}

}

};

#define REGISTER(Type) \

\

REGISTER\_KERNEL\_BUILDER( \

Name("BatchPredictionResponseWriter") \

.Device(DEVICE\_CPU) \

.TypeConstraint<Type>("T"), \

BatchPredictionResponseWriter<Type>); \

REGISTER(float);

REGISTER(double);