#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("BatchPredictionTensorResponseWriter")

.Attr("T: list({string, int32, int64, 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 dense tensors into a BatchPredictionResponse.

values: list of tensors

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

Outputs

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

)doc");

class BatchPredictionTensorResponseWriter : public OpKernel {

public:

explicit BatchPredictionTensorResponseWriter(OpKernelConstruction\* context)

: OpKernel(context) {}

void Compute(OpKernelContext\* context) override {

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

try {

// set keys as twml::Tensor

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

// check sizes

uint64\_t num\_keys = in\_keys\_.getNumElements();

uint64\_t num\_values = context->num\_inputs() - 1;

OP\_REQUIRES(context, num\_values % num\_keys == 0,

errors::InvalidArgument("Number of dense tensors not multiple of dense keys"));

// set dense tensor values

std::vector<twml::RawTensor> in\_values\_;

for (int i = 1; i < context->num\_inputs(); i++) {

in\_values\_.push\_back(TFTensor\_to\_twml\_raw\_tensor(context->input(i)));

}

// no continuous predictions in this op, only tensors

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

const twml::Tensor dummy\_cont\_values\_;

// call constructor BatchPredictionResponse

twml::BatchPredictionResponse tempResult(

dummy\_cont\_keys\_, dummy\_cont\_values\_, in\_keys\_, in\_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 = NULL;

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()));

}

}

};

REGISTER\_KERNEL\_BUILDER(

Name("BatchPredictionTensorResponseWriter").Device(DEVICE\_CPU),

BatchPredictionTensorResponseWriter);