#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"

#include "resource\_utils.h"

#include <algorithm>

using std::string;

template<typename IndexType, typename ValueType, bool calc\_batch\_size>

void ComputeFixedLengthTensor(OpKernelContext \*context, int64 max\_length\_) {

try {

const Tensor& segment\_ids = context->input(0);

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

const Tensor& pad\_value = context->input(2);

auto indices\_flat = segment\_ids.flat<IndexType>();

auto values\_flat = values.flat<ValueType>();

auto pad\_value\_scalar = pad\_value.scalar<ValueType>()();

// Get maximum length from batch if user hasn't specified it.

int64 max\_length = max\_length\_;

if (max\_length < 0 && indices\_flat.size() > 0) {

int64 current\_id = indices\_flat(0);

int64 current\_length = 1;

for (int64 i = 1; i < indices\_flat.size(); i++) {

if (current\_id == indices\_flat(i)) {

current\_length++;

} else {

current\_id = indices\_flat(i);

max\_length = std::max(max\_length, current\_length);

current\_length = 1;

}

}

// This is needed if the last batch is the longest sequence.

max\_length = std::max(max\_length, current\_length);

}

int64 batch\_size = 0;

if (calc\_batch\_size) {

if (indices\_flat.size() > 0) {

// The last value of segment\_ids will have value batch\_size 1;

batch\_size = 1 + indices\_flat(indices\_flat.size() - 1);

} else {

batch\_size = 0;

}

} else {

const Tensor& batch\_size\_tensor = context->input(3);

batch\_size = batch\_size\_tensor.flat<int64>()(0);

}

TensorShape output\_shape = {batch\_size, max\_length};

Tensor\* fixed\_length = nullptr;

OP\_REQUIRES\_OK(context, context->allocate\_output(0, output\_shape, &fixed\_length));

auto fixed\_length\_flat = fixed\_length->flat<ValueType>();

int64 n = 0;

int64 offset = 0;

for (int64 i = 0; i < batch\_size; i++) {

for (int64 j = 0; j < max\_length; j++) {

if (n < indices\_flat.size() && indices\_flat(n) == i) {

// Copy from variable length tensor.

fixed\_length\_flat(offset + j) = values\_flat(n);

n++;

} else {

// Pad to fixed length.

fixed\_length\_flat(offset + j) = pad\_value\_scalar;

}

}

// Corner case: truncate to max\_length if user specified max\_length < current length.

while (n < indices\_flat.size() && i == indices\_flat(n)) n++;

// Update output pointer

offset += max\_length;

}

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

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

}

}

REGISTER\_OP("FixedLengthTensor")

.Attr("IndexType: {int64, int32}")

.Attr("ValueType: {int64, int32, string}")

.Attr("max\_length: int")

.Input("segment\_ids: IndexType")

.Input("values: ValueType")

.Input("pad\_value: ValueType")

.Output("fixed\_length: ValueType")

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

return Status::OK();

}).Doc(R"doc(

A tensorflow OP to convert variable length segments into fixed length tensor.

Attr

max\_length: The size of the inner most (i.e. last) dimension.

Input

segment\_ids: 1D input tensor containing the sorted segment\_ids.

values: 1D input tensor containing the values.

pad\_value: The value used for padding the fixed length tensor.

Outputs

fixed\_length: A fixed length tensor of size [batch\_size, max\_length].

)doc");

template<typename IndexType, typename ValueType>

class FixedLengthTensor: public OpKernel {

public:

explicit FixedLengthTensor(OpKernelConstruction \*context) : OpKernel(context) {

OP\_REQUIRES\_OK(context, context->GetAttr("max\_length", &max\_length\_));

}

private:

int64 max\_length\_;

void Compute(OpKernelContext \*context) override {

ComputeFixedLengthTensor<IndexType, ValueType, true>(context, max\_length\_);

}

};

REGISTER\_OP("FixedLengthTensorV2")

.Attr("IndexType: {int64, int32}")

.Attr("ValueType: {int64, int32, string}")

.Attr("max\_length: int")

.Input("segment\_ids: IndexType")

.Input("values: ValueType")

.Input("pad\_value: ValueType")

.Input("batch\_size: int64")

.Output("fixed\_length: ValueType")

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

return Status::OK();

}).Doc(R"doc(

A tensorflow OP to convert variable length segments into fixed length tensor.

Attr

max\_length: The size of the inner most (i.e. last) dimension.

Input

segment\_ids: 1D input tensor containing the sorted segment\_ids.

values: 1D input tensor containing the values.

pad\_value: The value used for padding the fixed length tensor.

batch\_size: The batch size to use.

Outputs

fixed\_length: A fixed length tensor of size [batch\_size, max\_length].

)doc");

template<typename IndexType, typename ValueType>

class FixedLengthTensorV2: public OpKernel {

public:

explicit FixedLengthTensorV2(OpKernelConstruction \*context) : OpKernel(context) {

OP\_REQUIRES\_OK(context, context->GetAttr("max\_length", &max\_length\_));

}

private:

int64 max\_length\_;

void Compute(OpKernelContext \*context) override {

ComputeFixedLengthTensor<IndexType, ValueType, false>(context, max\_length\_);

}

};

#define REGISTER\_SPARSE\_TO\_FIXED\_LENGTH(IndexType, ValueType) \

REGISTER\_KERNEL\_BUILDER( \

Name("FixedLengthTensor") \

.Device(DEVICE\_CPU) \

.TypeConstraint<IndexType>("IndexType") \

.TypeConstraint<ValueType>("ValueType"), \

FixedLengthTensor<IndexType, ValueType>); \

\

REGISTER\_KERNEL\_BUILDER( \

Name("FixedLengthTensorV2") \

.Device(DEVICE\_CPU) \

.TypeConstraint<IndexType>("IndexType") \

.TypeConstraint<ValueType>("ValueType"), \

FixedLengthTensorV2<IndexType, ValueType>); \

REGISTER\_SPARSE\_TO\_FIXED\_LENGTH(int64, int64)

REGISTER\_SPARSE\_TO\_FIXED\_LENGTH(int64, int32)

REGISTER\_SPARSE\_TO\_FIXED\_LENGTH(int64, string)

REGISTER\_SPARSE\_TO\_FIXED\_LENGTH(int32, int64)

REGISTER\_SPARSE\_TO\_FIXED\_LENGTH(int32, int32)

REGISTER\_SPARSE\_TO\_FIXED\_LENGTH(int32, string)