#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 <string>

#include <set>

using std::string;

void join(const std::set<string>& v, char c, string& s) {

s.clear();

std::set<std::string>::iterator it = v.begin();

while (it != v.end()) {

s += \*it;

it++;

if (it != v.end()) s+= c;

}

}

// cpp function that computes substrings of a given word

std::string computeSubwords(std::string word, int32\_t minn, int32\_t maxn) {

std::string word2 = "<" + word + ">";

std::set<string> ngrams;

std::string s;

ngrams.insert(word);

ngrams.insert(word2);

for (size\_t i = 0; i < word2.size(); i++) {

if ((word2[i] & 0xC0) == 0x80) continue;

for (size\_t j = minn; i+j <= word2.size() && j <= maxn; j++) {

ngrams.insert(word2.substr(i, j));

}

}

join(ngrams, ';', s);

ngrams.clear();

return s;

}

// tf-op function that computes substrings for a given tensor of words

template< typename ValueType>

void ComputeSubStringsTensor(OpKernelContext \*context, int32 min\_n, int32 max\_n) {

try {

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

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

// batch\_size from input\_size :

const int batch\_size = values\_flat.size();

// define the output tensor

Tensor\* substrings = nullptr;

OP\_REQUIRES\_OK(context, context->allocate\_output(0, values.shape(), &substrings));

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

// compute substrings for the given tensor values

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

substrings\_flat(i) = computeSubwords(values\_flat(i), min\_n, max\_n);

}

}

catch (const std::exception &err) {

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

}

}

REGISTER\_OP("GetSubstrings")

.Attr("ValueType: {string}")

.Attr("min\_n: int")

.Attr("max\_n: int")

.Input("values: ValueType")

.Output("substrings: ValueType")

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

c->set\_output(0, c->input(0));

return Status::OK();

}).Doc(R"doc(

A tensorflow OP to convert word to substrings of length between min\_n and max\_n.

Attr

min\_n,max\_n: The size of the substrings.

Input

values: 1D input tensor containing the values.

Outputs

substrings: A string tensor where substrings are joined by ";".

)doc");

template<typename ValueType>

class GetSubstrings : public OpKernel {

public:

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

OP\_REQUIRES\_OK(context, context->GetAttr("min\_n", &min\_n));

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

}

private:

int32 min\_n;

int32 max\_n;

void Compute(OpKernelContext \*context) override {

ComputeSubStringsTensor<ValueType>(context, min\_n, max\_n);

}

};

#define REGISTER\_SUBSTRINGS(ValueType) \

REGISTER\_KERNEL\_BUILDER( \

Name("GetSubstrings") \

.Device(DEVICE\_CPU) \

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

GetSubstrings<ValueType>); \

REGISTER\_SUBSTRINGS(string)