#pragma once

#ifdef \_\_cplusplus

#include <twml/common.h>

#include <twml/defines.h>

#include <twml/HashedDataRecord.h>

#include <twml/TensorRecordReader.h>

#include <cstdint>

#include <vector>

#include <string>

#include <unordered\_map>

namespace twml {

enum class DecodeMode: int64\_t

{

hash\_valname = 0,

hash\_fname\_and\_valname = 1,

};

class TWMLAPI HashedDataRecordReader : public TensorRecordReader {

private:

typedef Map<int64\_t, int64\_t> KeyMap\_t;

KeyMap\_t \*m\_keep\_map;

KeyMap\_t \*m\_labels\_map;

KeyMap\_t \*m\_weights\_map;

DecodeMode m\_decode\_mode;

public:

bool keepId (const int64\_t &key, int64\_t &code);

bool isLabel (const int64\_t &key, int64\_t &code);

bool isWeight (const int64\_t &key, int64\_t &code);

void readBinary (const int feature\_type , HashedDataRecord \*record);

void readContinuous (const int feature\_type , HashedDataRecord \*record);

void readDiscrete (const int feature\_type , HashedDataRecord \*record);

void readString (const int feature\_type , HashedDataRecord \*record);

void readSparseBinary (const int feature\_type , HashedDataRecord \*record);

void readSparseContinuous (const int feature\_type , HashedDataRecord \*record);

void readBlob (const int feature\_type , HashedDataRecord \*record);

HashedDataRecordReader() :

TensorRecordReader(nullptr),

m\_keep\_map(nullptr),

m\_labels\_map(nullptr),

m\_weights\_map(nullptr),

m\_decode\_mode(DecodeMode::hash\_valname)

{}

// Using a template instead of int64\_t because tensorflow implements int64 based on compiler.

void setKeepMap(KeyMap\_t \*keep\_map) {

m\_keep\_map = keep\_map;

}

void setLabelsMap(KeyMap\_t \*labels\_map) {

m\_labels\_map = labels\_map;

}

void setWeightsMap(KeyMap\_t \*weights\_map) {

m\_weights\_map = weights\_map;

}

void setDecodeMode(int64\_t mode) {

m\_decode\_mode = static\_cast<DecodeMode>(mode);

}

};

}

#endif