#pragma once

#ifdef \_\_cplusplus

#include <twml/defines.h>

#include <twml/TensorRecord.h>

#include <cstdint>

#include <cmath>

#include <vector>

namespace twml {

class HashedDataRecordReader;

class TWMLAPI HashedDataRecord : public TensorRecord {

public:

typedef HashedDataRecordReader Reader;

HashedDataRecord(int num\_labels=0, int num\_weights=0):

m\_keys(),

m\_transformed\_keys(),

m\_values(),

m\_codes(),

m\_types(),

m\_labels(num\_labels, std::nanf("")),

m\_weights(num\_weights) {}

void decode(HashedDataRecordReader &reader);

const std::vector<int64\_t> &keys() const { return m\_keys; }

const std::vector<int64\_t> &transformed\_keys() const { return m\_transformed\_keys; }

const std::vector<double> &values() const { return m\_values; }

const std::vector<int64\_t> &codes() const { return m\_codes; }

const std::vector<uint8\_t> &types() const { return m\_types; }

const std::vector<float> &labels() const { return m\_labels; }

const std::vector<float> &weights() const { return m\_weights; }

void clear();

uint64\_t totalSize() const { return m\_keys.size(); }

void extendSize(int delta\_size) {

int count = m\_keys.size() + delta\_size;

m\_keys.reserve(count);

m\_transformed\_keys.reserve(count);

m\_values.reserve(count);

m\_codes.reserve(count);

m\_types.reserve(count);

}

private:

std::vector<int64\_t> m\_keys;

std::vector<int64\_t> m\_transformed\_keys;

std::vector<double> m\_values;

std::vector<int64\_t> m\_codes;

std::vector<uint8\_t> m\_types;

std::vector<float> m\_labels;

std::vector<float> m\_weights;

void addKey(int64\_t key, int64\_t transformed\_key, int64\_t code, uint8\_t type, double value=1);

void addLabel(int64\_t id, double value = 1);

void addWeight(int64\_t id, double value);

friend class HashedDataRecordReader;

};

}

#endif