#pragma once

#ifdef \_\_cplusplus

#include <twml/common.h>

#include <twml/defines.h>

#include <twml/TensorRecord.h>

#include <cstdint>

#include <cmath>

#include <string>

#include <unordered\_map>

#include <unordered\_set>

#include <vector>

namespace twml {

class DataRecordReader;

class TWMLAPI DataRecord : public TensorRecord {

public:

typedef std::vector<std::pair<std::string, double>> SparseContinuousValueType;

typedef std::vector<std::string> SparseBinaryValueType;

typedef Set<int64\_t> BinaryFeatures;

typedef Map<int64\_t, double> ContinuousFeatures;

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

typedef Map<int64\_t, std::string> StringFeatures;

typedef Map<int64\_t, SparseBinaryValueType> SparseBinaryFeatures;

typedef Map<int64\_t, SparseContinuousValueType> SparseContinuousFeatures;

typedef Map<int64\_t, std::vector<uint8\_t>> BlobFeatures;

private:

BinaryFeatures m\_binary;

ContinuousFeatures m\_continuous;

DiscreteFeatures m\_discrete;

StringFeatures m\_string;

SparseBinaryFeatures m\_sparsebinary;

SparseContinuousFeatures m\_sparsecontinuous;

BlobFeatures m\_blob;

std::vector<float> m\_labels;

std::vector<float> m\_weights;

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

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

public:

typedef DataRecordReader Reader;

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

m\_binary(),

m\_continuous(),

m\_discrete(),

m\_string(),

m\_sparsebinary(),

m\_sparsecontinuous(),

m\_blob(),

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

m\_weights(num\_weights) {

#ifdef USE\_DENSE\_HASH

m\_binary.set\_empty\_key(0);

m\_continuous.set\_empty\_key(0);

m\_discrete.set\_empty\_key(0);

m\_string.set\_empty\_key(0);

m\_sparsebinary.set\_empty\_key(0);

m\_sparsecontinuous.set\_empty\_key(0);

#endif

m\_binary.max\_load\_factor(0.5);

m\_continuous.max\_load\_factor(0.5);

m\_discrete.max\_load\_factor(0.5);

m\_string.max\_load\_factor(0.5);

m\_sparsebinary.max\_load\_factor(0.5);

m\_sparsecontinuous.max\_load\_factor(0.5);

}

const BinaryFeatures &getBinary() const { return m\_binary; }

const ContinuousFeatures &getContinuous() const { return m\_continuous; }

const DiscreteFeatures &getDiscrete() const { return m\_discrete; }

const StringFeatures &getString() const { return m\_string; }

const SparseBinaryFeatures &getSparseBinary() const { return m\_sparsebinary; }

const SparseContinuousFeatures &getSparseContinuous() const { return m\_sparsecontinuous; }

const BlobFeatures &getBlob() const { return m\_blob; }

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

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

// used by DataRecordWriter

template <typename T>

void addContinuous(std::vector<int64\_t> feature\_ids, std::vector<T> values) {

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

m\_continuous[feature\_ids[i]] = values[i];

}

}

template <typename T>

void addContinuous(const int64\_t \*keys, uint64\_t num\_keys, T \*values) {

for (size\_t i = 0; i < num\_keys; ++i){

m\_continuous[keys[i]] = values[i];

}

}

void decode(DataRecordReader &reader);

void clear();

friend class DataRecordReader;

};

}

#endif