#include "internal/thrift.h"

#include "internal/error.h"

#include <twml/utilities.h>

#include <twml/DataRecord.h>

#include <twml/DataRecordReader.h>

#include <twml/Error.h>

#include <cstring>

#include <cstdint>

namespace twml {

void DataRecord::decode(DataRecordReader &reader) {

uint8\_t feature\_type = reader.readByte();

while (feature\_type != TTYPE\_STOP) {

int16\_t field\_id = reader.readInt16();

switch (field\_id) {

case DR\_BINARY:

reader.readBinary(feature\_type, this);

break;

case DR\_CONTINUOUS:

reader.readContinuous(feature\_type, this);

break;

case DR\_DISCRETE:

reader.readDiscrete(feature\_type, this);

break;

case DR\_STRING:

reader.readString(feature\_type, this);

break;

case DR\_SPARSE\_BINARY:

reader.readSparseBinary(feature\_type, this);

break;

case DR\_SPARSE\_CONTINUOUS:

reader.readSparseContinuous(feature\_type, this);

break;

case DR\_BLOB:

reader.readBlob(feature\_type, this);

break;

case DR\_GENERAL\_TENSOR:

reader.readTensor(feature\_type, dynamic\_cast<TensorRecord \*>(this));

break;

case DR\_SPARSE\_TENSOR:

reader.readSparseTensor(feature\_type, dynamic\_cast<TensorRecord \*>(this));

break;

default:

throw ThriftInvalidField(field\_id, "DataRecord::decode");

}

feature\_type = reader.readByte();

}

}

void DataRecord::addLabel(int64\_t id, double label) {

m\_labels[id] = label;

}

void DataRecord::addWeight(int64\_t id, double val) {

m\_weights[id] = val;

}

void DataRecord::clear() {

std::fill(m\_labels.begin(), m\_labels.end(), std::nanf(""));

std::fill(m\_weights.begin(), m\_weights.end(), 0.0);

m\_binary.clear();

m\_continuous.clear();

m\_discrete.clear();

m\_string.clear();

m\_sparsebinary.clear();

m\_sparsecontinuous.clear();

}

} // namespace twml