package com.twitter.search.earlybird.search.relevance.scoring;

import java.nio.ByteBuffer;

import java.nio.ByteOrder;

// Ideally, this part should live somewhere in the Cortex common

// code. Today, it is not possible to create

// a `SparseTensor` that relies only on ByteBuffer.

public class SparseTensor {

private ByteBuffer sparseIndices;

private ByteBuffer sparseValues;

private ByteBuffer sparseShape;

private int numDocs;

private final long[] sparseShapeShapeDimension = new long[] {2L};

private final long inputBitSize = 1 << 63;

private long numRecordsSeen = 0;

private final long numFeatures;

private int numValuesSeen;

public SparseTensor(int numDocs, int numFeatures) {

this.numDocs = numDocs;

this.numFeatures = (long) numFeatures;

this.sparseValues =

ByteBuffer

.allocate(numFeatures \* numDocs \* Float.BYTES)

.order(ByteOrder.LITTLE\_ENDIAN);

this.sparseIndices =

ByteBuffer

.allocate(2 \* numFeatures \* numDocs \* Long.BYTES)

.order(ByteOrder.LITTLE\_ENDIAN);

this.sparseShape =

ByteBuffer

.allocate(2 \* Long.BYTES)

.order(ByteOrder.LITTLE\_ENDIAN);

}

public void incNumRecordsSeen() {

numRecordsSeen++;

}

/\*\*

\* Adds the given value to this tensor.

\*/

public void addValue(long featureId, float value) {

sparseValues.putFloat(value);

sparseIndices.putLong(numRecordsSeen);

sparseIndices.putLong(featureId);

numValuesSeen++;

}

public ByteBuffer getSparseValues() {

sparseValues.limit(numValuesSeen \* Float.BYTES);

sparseValues.rewind();

return sparseValues;

}

public long[] getSparseValuesShape() {

return new long[] {numValuesSeen};

}

public long[] getSparseIndicesShape() {

return new long[] {numValuesSeen, 2L};

}

public long[] getSparseShapeShape() {

return sparseShapeShapeDimension;

}

public ByteBuffer getSparseIndices() {

sparseIndices.limit(2 \* numValuesSeen \* Long.BYTES);

sparseIndices.rewind();

return sparseIndices;

}

/\*\*

\* Returns the sparse shape for this tensor.

\*/

public ByteBuffer getSparseShape() {

sparseShape.putLong(numRecordsSeen);

sparseShape.putLong(inputBitSize);

sparseShape.rewind();

return sparseShape;

}

}