package com.twitter.search.core.earlybird.facets;

import com.google.common.base.Preconditions;

import org.apache.lucene.facet.FacetResult;

import org.apache.lucene.facet.LabelAndValue;

import org.apache.lucene.util.BytesRef;

import org.apache.lucene.util.PriorityQueue;

import com.twitter.search.common.facets.FacetSearchParam;

import com.twitter.search.core.earlybird.facets.FacetLabelProvider.FacetLabelAccessor;

import it.unimi.dsi.fastutil.ints.Int2IntMap.Entry;

import it.unimi.dsi.fastutil.ints.Int2IntMap.FastEntrySet;

import it.unimi.dsi.fastutil.ints.Int2IntOpenHashMap;

public class PerfieldFacetCountAggregator {

private final Int2IntOpenHashMap countMap;

private final FacetLabelAccessor facetLabelAccessor;

private final String name;

/\*\*

\* Creates a new per-field facet aggregator.

\*/

public PerfieldFacetCountAggregator(String name, FacetLabelProvider facetLabelProvider) {

this.name = name;

this.countMap = new Int2IntOpenHashMap();

this.countMap.defaultReturnValue(0);

this.facetLabelAccessor = facetLabelProvider.getLabelAccessor();

}

public void collect(int termId) {

countMap.put(termId, countMap.get(termId) + 1);

}

/\*\*

\* Returns the top facets.

\*/

public FacetResult getTop(FacetSearchParam facetSearchParam) {

Preconditions.checkArgument(

facetSearchParam != null

&& facetSearchParam.getFacetFieldRequest().getField().equals(name)

&& (facetSearchParam.getFacetFieldRequest().getPath() == null

|| facetSearchParam.getFacetFieldRequest().getPath().isEmpty()));

PriorityQueue<Entry> pq = new PriorityQueue<Entry>(

facetSearchParam.getFacetFieldRequest().getNumResults()) {

private BytesRef buffer = new BytesRef();

@Override

protected boolean lessThan(Entry a, Entry b) {

// first by count desc

int r = Integer.compare(a.getIntValue(), b.getIntValue());

if (r != 0) {

return r < 0;

}

// and then by label asc

BytesRef label1 = facetLabelAccessor.getTermRef(a.getIntKey());

buffer.bytes = label1.bytes;

buffer.offset = label1.offset;

buffer.length = label1.length;

return buffer.compareTo(facetLabelAccessor.getTermRef(b.getIntKey())) > 0;

}

};

final FastEntrySet entrySet = countMap.int2IntEntrySet();

int numValid = 0;

for (Entry entry : entrySet) {

long val = entry.getIntValue();

if (val > 0) {

numValid++;

pq.insertWithOverflow(entry);

}

}

int numVals = pq.size();

LabelAndValue[] labelValues = new LabelAndValue[numVals];

// Priority queue pops out "least" element first (that is the root).

// Least in our definition regardless of how we define what that is should be the last element.

for (int i = labelValues.length - 1; i >= 0; i--) {

Entry entry = pq.pop();

labelValues[i] = new LabelAndValue(

facetLabelAccessor.getTermText(entry.getIntKey()),

entry.getValue());

}

return new FacetResult(name, null, 0, labelValues, numValid);

}

}