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

import java.io.IOException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.PriorityQueue;

import com.google.common.base.Preconditions;

import com.twitter.common.util.Clock;

import com.twitter.search.common.constants.thriftjava.ThriftLanguage;

import com.twitter.search.common.ranking.thriftjava.ThriftFacetEarlybirdSortingMode;

import com.twitter.search.common.schema.base.ImmutableSchemaInterface;

import com.twitter.search.core.earlybird.facets.DummyFacetAccumulator;

import com.twitter.search.core.earlybird.facets.FacetAccumulator;

import com.twitter.search.core.earlybird.facets.FacetCountIterator;

import com.twitter.search.core.earlybird.facets.FacetIDMap;

import com.twitter.search.core.earlybird.facets.FacetIDMap.FacetField;

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

import com.twitter.search.core.earlybird.facets.LanguageHistogram;

import com.twitter.search.core.earlybird.index.EarlybirdIndexSegmentAtomicReader;

import com.twitter.search.earlybird.search.AbstractResultsCollector;

import com.twitter.search.earlybird.search.AntiGamingFilter;

import com.twitter.search.earlybird.search.EarlybirdLuceneSearcher.FacetSearchResults;

import com.twitter.search.earlybird.stats.EarlybirdSearcherStats;

import com.twitter.search.earlybird.thrift.ThriftFacetCount;

import com.twitter.search.earlybird.thrift.ThriftFacetFieldResults;

public class FacetResultsCollector extends

AbstractResultsCollector<FacetSearchRequestInfo, FacetSearchResults> {

private final FacetScorer facetScorer;

private final ThriftFacetEarlybirdSortingMode sortingMode;

static class Accumulator {

protected final FacetAccumulator<ThriftFacetFieldResults>[] accumulators;

protected final FacetCountIterator accessor;

protected final FacetIDMap facetIDMap;

Accumulator(FacetAccumulator<ThriftFacetFieldResults>[] accumulators,

FacetCountIterator accessor,

FacetIDMap facetIDMap) {

this.accumulators = accumulators;

this.accessor = accessor;

this.facetIDMap = facetIDMap;

}

FacetAccumulator<ThriftFacetFieldResults> getFacetAccumulator(String facetName) {

FacetField facet = facetIDMap.getFacetFieldByFacetName(facetName);

return accumulators[facet.getFacetId()];

}

}

private Accumulator currentAccumulator;

private List<Accumulator> segAccumulators;

private final HashingAndPruningFacetAccumulator.FacetComparator facetComparator;

/\*\*

\* Creates a new FacetResultsCollector for the given facet search request.

\*/

public FacetResultsCollector(

ImmutableSchemaInterface schema,

FacetSearchRequestInfo searchRequestInfo,

AntiGamingFilter antiGamingFilter,

EarlybirdSearcherStats searcherStats,

Clock clock,

int requestDebugInfo) {

super(schema, searchRequestInfo, clock, searcherStats, requestDebugInfo);

if (searchRequestInfo.rankingOptions != null

&& searchRequestInfo.rankingOptions.isSetSortingMode()) {

this.sortingMode = searchRequestInfo.rankingOptions.getSortingMode();

} else {

this.sortingMode = ThriftFacetEarlybirdSortingMode.SORT\_BY\_WEIGHTED\_COUNT;

}

this.facetComparator = HashingAndPruningFacetAccumulator.getComparator(sortingMode);

this.facetScorer = createScorer(antiGamingFilter);

this.segAccumulators = new ArrayList<>();

}

@Override

public void startSegment() {

currentAccumulator = null;

}

@Override

public void doCollect(long tweetID) throws IOException {

if (currentAccumulator == null) {

// Lazily create accumulators. Most segment / query / facet combinations have no hits.

currentAccumulator = newPerSegmentAccumulator(currTwitterReader);

segAccumulators.add(currentAccumulator);

facetScorer.startSegment(currTwitterReader);

}

facetScorer.incrementCounts(currentAccumulator, curDocId);

}

@Override

public FacetSearchResults doGetResults() {

return new FacetSearchResults(this);

}

/\*\*

\* Returns the top-k facet results for the requested facetName.

\*/

public ThriftFacetFieldResults getFacetResults(String facetName, int topK) {

int totalCount = 0;

final Map<String, ThriftFacetCount> map = new HashMap<>();

LanguageHistogram languageHistogram = new LanguageHistogram();

for (Accumulator segAccumulator : segAccumulators) {

FacetAccumulator<ThriftFacetFieldResults> accumulator =

segAccumulator.getFacetAccumulator(facetName);

Preconditions.checkNotNull(accumulator);

ThriftFacetFieldResults results = accumulator.getAllFacets();

if (results == null) {

continue;

}

totalCount += results.totalCount;

// merge language histograms from different segments

languageHistogram.addAll(accumulator.getLanguageHistogram());

for (ThriftFacetCount facetCount : results.getTopFacets()) {

String label = facetCount.getFacetLabel();

ThriftFacetCount oldCount = map.get(label);

if (oldCount != null) {

oldCount.setSimpleCount(oldCount.getSimpleCount() + facetCount.getSimpleCount());

oldCount.setWeightedCount(oldCount.getWeightedCount() + facetCount.getWeightedCount());

oldCount.setFacetCount(oldCount.getFacetCount() + facetCount.getFacetCount());

oldCount.setPenaltyCount(oldCount.getPenaltyCount() + facetCount.getPenaltyCount());

} else {

map.put(label, facetCount);

}

}

}

if (map.size() == 0 || totalCount == 0) {

// No results.

return null;

}

// sort table wrt percentage

PriorityQueue<ThriftFacetCount> pq =

new PriorityQueue<>(map.size(), facetComparator.getThriftComparator(true));

pq.addAll(map.values());

ThriftFacetFieldResults results = new ThriftFacetFieldResults();

results.setTopFacets(new ArrayList<>());

results.setTotalCount(totalCount);

// Store merged language histogram into thrift object

for (Map.Entry<ThriftLanguage, Integer> entry

: languageHistogram.getLanguageHistogramAsMap().entrySet()) {

results.putToLanguageHistogram(entry.getKey(), entry.getValue());

}

// Get top facets.

for (int i = 0; i < topK && i < map.size(); i++) {

ThriftFacetCount facetCount = pq.poll();

if (facetCount != null) {

results.addToTopFacets(facetCount);

}

}

return results;

}

protected FacetScorer createScorer(AntiGamingFilter antiGamingFilter) {

if (searchRequestInfo.rankingOptions != null) {

return new DefaultFacetScorer(searchRequestInfo.getSearchQuery(),

searchRequestInfo.rankingOptions,

antiGamingFilter,

sortingMode);

} else {

return new FacetScorer() {

@Override

protected void startSegment(EarlybirdIndexSegmentAtomicReader reader) {

}

@Override

public void incrementCounts(Accumulator accumulator, int internalDocID) throws IOException {

accumulator.accessor.incrementData.accumulators = accumulator.accumulators;

accumulator.accessor.incrementData.weightedCountIncrement = 1;

accumulator.accessor.incrementData.penaltyIncrement = 0;

accumulator.accessor.incrementData.languageId = ThriftLanguage.UNKNOWN.getValue();

accumulator.accessor.collect(internalDocID);

}

@Override

public FacetAccumulator getFacetAccumulator(FacetLabelProvider labelProvider) {

return new HashingAndPruningFacetAccumulator(labelProvider, facetComparator);

}

};

}

}

protected Accumulator newPerSegmentAccumulator(EarlybirdIndexSegmentAtomicReader indexReader) {

final FacetIDMap facetIDMap = indexReader.getFacetIDMap();

final FacetCountIterator accessor =

indexReader.getFacetCountingArray().getIterator(

indexReader,

getSearchRequestInfo().getFacetCountState(),

TweetSearchFacetCountIteratorFactory.FACTORY);

final FacetAccumulator<ThriftFacetFieldResults>[] accumulators =

(FacetAccumulator<ThriftFacetFieldResults>[])

new FacetAccumulator[facetIDMap.getNumberOfFacetFields()];

Map<String, FacetLabelProvider> labelProviders = indexReader.getFacetLabelProviders();

for (FacetField f : facetIDMap.getFacetFields()) {

int id = f.getFacetId();

if (getSearchRequestInfo().getFacetCountState().isCountField(f.getFieldInfo())) {

accumulators[id] = (FacetAccumulator<ThriftFacetFieldResults>) facetScorer

.getFacetAccumulator(labelProviders.get(f.getFacetName()));

} else {

// Dummmy accumulator does nothing.

accumulators[id] = new DummyFacetAccumulator();

}

}

return new Accumulator(accumulators, accessor, facetIDMap);

}

}