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 com.google.common.collect.Maps;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.twitter.common.collections.Pair;

import com.twitter.common.util.Clock;

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

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

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

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

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

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

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

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

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

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

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

public class ExplainFacetResultsCollector extends FacetResultsCollector {

private static final Logger LOG =

LoggerFactory.getLogger(ExplainFacetResultsCollector.class.getName());

protected final List<Pair<Integer, Long>> proofs;

protected final Map<String, Map<String, List<Long>>> proofAccumulators;

protected Map<String, FacetLabelProvider> facetLabelProviders;

private FacetIDMap facetIDMap;

/\*\*

\* Creates a new facet collector with the ability to provide explanations for the search results.

\*/

public ExplainFacetResultsCollector(

ImmutableSchemaInterface schema,

FacetSearchRequestInfo searchRequestInfo,

AntiGamingFilter antiGamingFilter,

EarlybirdSearcherStats searcherStats,

Clock clock,

int requestDebugMode) throws IOException {

super(schema, searchRequestInfo, antiGamingFilter, searcherStats, clock, requestDebugMode);

proofs = new ArrayList<>(128);

proofAccumulators = Maps.newHashMap();

for (Schema.FieldInfo facetField : schema.getFacetFields()) {

HashMap<String, List<Long>> fieldLabelToTweetIdsMap = new HashMap<>();

proofAccumulators.put(facetField.getFieldType().getFacetName(), fieldLabelToTweetIdsMap);

}

}

@Override

protected Accumulator newPerSegmentAccumulator(EarlybirdIndexSegmentAtomicReader indexReader) {

Accumulator accumulator = super.newPerSegmentAccumulator(indexReader);

accumulator.accessor.setProofs(proofs);

facetLabelProviders = indexReader.getFacetLabelProviders();

facetIDMap = indexReader.getFacetIDMap();

return accumulator;

}

@Override

public void doCollect(long tweetID) throws IOException {

proofs.clear();

// FacetResultsCollector.doCollect() calls FacetScorer.incrementCounts(),

// FacetResultsCollector.doCollect() creates a FacetResultsCollector.Accumulator, if

// necessary, which contains the accessor (a CompositeFacetIterator) and accumulators

// (FacetAccumulator of each field)

super.doCollect(tweetID);

for (Pair<Integer, Long> fieldIdTermIdPair : proofs) {

int fieldID = fieldIdTermIdPair.getFirst();

long termID = fieldIdTermIdPair.getSecond();

// Convert term ID to the term text, a.k.a. facet label

String facetName = facetIDMap.getFacetFieldByFacetID(fieldID).getFacetName();

if (facetName != null) {

String facetLabel = facetLabelProviders.get(facetName)

.getLabelAccessor().getTermText(termID);

List<Long> tweetIDs = proofAccumulators.get(facetName).get(facetLabel);

if (tweetIDs == null) {

tweetIDs = new ArrayList<>();

proofAccumulators.get(facetName).put(facetLabel, tweetIDs);

}

tweetIDs.add(tweetID);

}

}

// clear it again just to be sure

proofs.clear();

}

/\*\*

\* Sets explanations for the facet results.

\*/

public void setExplanations(ThriftFacetResults facetResults) {

StringBuilder explanation = new StringBuilder();

for (Map.Entry<String, ThriftFacetFieldResults> facetFieldResultsEntry

: facetResults.getFacetFields().entrySet()) {

String facetName = facetFieldResultsEntry.getKey();

ThriftFacetFieldResults facetFieldResults = facetFieldResultsEntry.getValue();

Map<String, List<Long>> proofAccumulator = proofAccumulators.get(facetName);

if (proofAccumulator == null) {

// did not accumulate explanation for this facet type? a bug?

LOG.warn("No explanation accumulated for facet type " + facetName);

continue;

}

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

String facetLabel = facetCount.getFacetLabel(); // a.k.a. term text

ThriftFacetCountMetadata metadata = facetCount.getMetadata();

List<Long> tweetIDs = proofAccumulator.get(facetLabel);

if (tweetIDs == null) {

// did not accumulate explanation for this facet label? a bug?

LOG.warn("No explanation accumulated for " + facetLabel + " of facet type " + facetName);

continue;

}

explanation.setLength(0);

String oldExplanation = null;

if (metadata.isSetExplanation()) {

// save the old explanation from TwitterInMemoryIndexSearcher.fillTermMetadata()

oldExplanation = metadata.getExplanation();

// as of 2012/05/29, we have 18 digits tweet IDs

explanation.ensureCapacity(oldExplanation.length() + (18 + 2) + 10);

} else {

// as of 2012/05/29, we have 18 digits tweet IDs

explanation.ensureCapacity(tweetIDs.size() \* (18 + 2) + 10);

}

explanation.append("[");

for (Long tweetID : tweetIDs) {

explanation.append(tweetID)

.append(", ");

}

explanation.setLength(explanation.length() - 2); // remove the last ", "

explanation.append("]\n");

if (oldExplanation != null) {

explanation.append(oldExplanation);

}

metadata.setExplanation(explanation.toString());

}

}

}

}