package com.twitter.search.earlybird\_root.mergers;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collections;

import java.util.HashMap;

import java.util.HashSet;

import java.util.List;

import java.util.Map;

import java.util.Set;

import java.util.concurrent.TimeUnit;

import com.google.common.collect.Sets;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.twitter.search.common.logging.DebugMessageBuilder;

import com.twitter.search.common.metrics.SearchTimerStats;

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

import com.twitter.search.common.schema.earlybird.EarlybirdFieldConstants.EarlybirdFieldConstant;

import com.twitter.search.common.util.earlybird.FacetsResultsUtils;

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

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

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;

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

import com.twitter.search.earlybird\_root.common.EarlybirdRequestContext;

import com.twitter.util.Future;

/\*\*

\* Merger class to merge facets EarlybirdResponse objects

\*/

public class FacetResponseMerger extends EarlybirdResponseMerger {

private static final Logger LOG = LoggerFactory.getLogger(FacetResponseMerger.class);

private static final SearchTimerStats TIMER =

SearchTimerStats.export("merge\_facets", TimeUnit.NANOSECONDS, false, true);

private static final double SUCCESSFUL\_RESPONSE\_THRESHOLD = 0.9;

private final DebugMessageBuilder debugMessageBuilder;

/\*\*

\* Constructor to create the merger

\*/

public FacetResponseMerger(EarlybirdRequestContext requestContext,

List<Future<EarlybirdResponse>> responses,

ResponseAccumulator mode) {

super(requestContext, responses, mode);

debugMessageBuilder = responseMessageBuilder.getDebugMessageBuilder();

debugMessageBuilder.verbose("--- Request Received: %s", requestContext.getRequest());

}

@Override

protected SearchTimerStats getMergedResponseTimer() {

return TIMER;

}

@Override

protected double getDefaultSuccessResponseThreshold() {

return SUCCESSFUL\_RESPONSE\_THRESHOLD;

}

@Override

protected EarlybirdResponse internalMerge(EarlybirdResponse facetsResponse) {

final Map<String, FacetsResultsUtils.FacetFieldInfo> facetFieldInfoMap =

new HashMap<>();

final Set<Long> userIDWhitelist = new HashSet<>();

// First, parse the responses and build up our facet info map.

boolean termStatsFilteringMode = FacetsResultsUtils.prepareFieldInfoMap(

requestContext.getRequest().getFacetRequest(), facetFieldInfoMap);

// Iterate through all futures and get results.

collectResponsesAndPopulateMap(facetFieldInfoMap, userIDWhitelist);

// Next, aggregate the top facets and update the blender response.

facetsResponse

.setFacetResults(new ThriftFacetResults()

.setFacetFields(new HashMap<>())

.setUserIDWhitelist(userIDWhitelist));

// keep track of how many facets a user contributed - this map gets reset for every field

Map<Long, Integer> perFieldAntiGamingMap = new HashMap<>();

// this one is used for images and twimges

Map<Long, Integer> imagesAntiGamingMap = new HashMap<>();

Set<String> twimgDedupSet = null;

for (final Map.Entry<String, FacetsResultsUtils.FacetFieldInfo> entry

: facetFieldInfoMap.entrySet()) {

// reset for each field

String field = entry.getKey();

final Map<Long, Integer> antiGamingMap;

if (field.equals(EarlybirdFieldConstant.IMAGES\_FACET)

|| field.equals(EarlybirdFieldConstant.TWIMG\_FACET)) {

antiGamingMap = imagesAntiGamingMap;

} else {

perFieldAntiGamingMap.clear();

antiGamingMap = perFieldAntiGamingMap;

}

ThriftFacetFieldResults results = new ThriftFacetFieldResults();

FacetsResultsUtils.FacetFieldInfo info = entry.getValue();

results.setTotalCount(info.totalCounts);

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

FacetsResultsUtils.fillTopLanguages(info, results);

if (info.topFacets != null && !info.topFacets.isEmpty()) {

fillFacetFieldResults(info, antiGamingMap, results);

}

if (field.equals(EarlybirdFieldConstant.TWIMG\_FACET)) {

if (twimgDedupSet == null) {

twimgDedupSet = Sets.newHashSet();

}

FacetsResultsUtils.dedupTwimgFacet(twimgDedupSet, results, debugMessageBuilder);

}

facetsResponse.getFacetResults().putToFacetFields(entry.getKey(), results);

}

if (!termStatsFilteringMode) {

// in term stats filtering mode, if doing it here would break term stats filtering

FacetsResultsUtils.mergeTwimgResults(

facetsResponse.getFacetResults(),

Collections.<ThriftFacetCount>reverseOrder(

FacetsResultsUtils.getFacetCountComparator(

requestContext.getRequest().getFacetRequest())));

}

// Update the numHitsProcessed on ThriftSearchResults.

int numHitsProcessed = 0;

int numPartitionsEarlyTerminated = 0;

for (EarlybirdResponse earlybirdResponse: accumulatedResponses.getSuccessResponses()) {

ThriftSearchResults searchResults = earlybirdResponse.getSearchResults();

if (searchResults != null) {

numHitsProcessed += searchResults.getNumHitsProcessed();

numPartitionsEarlyTerminated += searchResults.getNumPartitionsEarlyTerminated();

}

}

ThriftSearchResults searchResults = new ThriftSearchResults();

searchResults.setResults(new ArrayList<>()); // required field

searchResults.setNumHitsProcessed(numHitsProcessed);

searchResults.setNumPartitionsEarlyTerminated(numPartitionsEarlyTerminated);

facetsResponse.setSearchResults(searchResults);

LOG.debug("Facets call completed successfully: {}", facetsResponse);

FacetsResultsUtils.fixNativePhotoUrl(facetsResponse);

return facetsResponse;

}

private void fillFacetFieldResults(FacetsResultsUtils.FacetFieldInfo facetFieldInfo,

Map<Long, Integer> antiGamingMap,

ThriftFacetFieldResults results) {

int minWeightedCount = 0;

int minSimpleCount = 0;

int maxPenaltyCount = Integer.MAX\_VALUE;

double maxPenaltyCountRatio = 1;

boolean excludePossiblySensitiveFacets = false;

boolean onlyReturnFacetsWithDisplayTweet = false;

int maxHitsPerUser = -1;

EarlybirdRequest request = requestContext.getRequest();

if (request.getFacetRequest() != null) {

ThriftFacetRankingOptions rankingOptions = request.getFacetRequest().getFacetRankingOptions();

if (request.getSearchQuery() != null) {

maxHitsPerUser = request.getSearchQuery().getMaxHitsPerUser();

}

if (rankingOptions != null) {

LOG.debug("FacetsResponseMerger: Using rankingOptions={}", rankingOptions);

if (rankingOptions.isSetMinCount()) {

minWeightedCount = rankingOptions.getMinCount();

}

if (rankingOptions.isSetMinSimpleCount()) {

minSimpleCount = rankingOptions.getMinSimpleCount();

}

if (rankingOptions.isSetMaxPenaltyCount()) {

maxPenaltyCount = rankingOptions.getMaxPenaltyCount();

}

if (rankingOptions.isSetMaxPenaltyCountRatio()) {

maxPenaltyCountRatio = rankingOptions.getMaxPenaltyCountRatio();

}

if (rankingOptions.isSetExcludePossiblySensitiveFacets()) {

excludePossiblySensitiveFacets = rankingOptions.isExcludePossiblySensitiveFacets();

}

if (rankingOptions.isSetOnlyReturnFacetsWithDisplayTweet()) {

onlyReturnFacetsWithDisplayTweet = rankingOptions.isOnlyReturnFacetsWithDisplayTweet();

}

}

} else {

LOG.warn("earlybirdRequest.getFacetRequest() is null");

}

ThriftFacetCount[] topFacetsArray = new ThriftFacetCount[facetFieldInfo.topFacets.size()];

facetFieldInfo.topFacets.values().toArray(topFacetsArray);

Arrays.sort(topFacetsArray, Collections.<ThriftFacetCount>reverseOrder(

FacetsResultsUtils.getFacetCountComparator(request.getFacetRequest())));

int numResults = capFacetFieldWidth(facetFieldInfo.fieldRequest.numResults);

if (topFacetsArray.length < numResults) {

numResults = topFacetsArray.length;

}

int collected = 0;

for (int i = 0; i < topFacetsArray.length; ++i) {

ThriftFacetCount count = topFacetsArray[i];

if (onlyReturnFacetsWithDisplayTweet

&& (!count.isSetMetadata() || !count.getMetadata().isSetStatusId()

|| count.getMetadata().getStatusId() == -1)) {

// status id must be set

continue;

}

if (excludePossiblySensitiveFacets && count.isSetMetadata()

&& count.getMetadata().isStatusPossiblySensitive()) {

// the display tweet may be offensive or NSFW

if (DebugMessageBuilder.DEBUG\_VERBOSE <= debugMessageBuilder.getDebugLevel()) {

debugMessageBuilder.verbose2("[%d] FacetsResponseMerger EXCLUDED: offensive or NSFW %s, "

+ "explanation: %s",

i, facetCountSummary(count),

count.getMetadata().getExplanation());

}

continue;

}

boolean filterOutUser = false;

if (maxHitsPerUser != -1 && count.isSetMetadata()) {

ThriftFacetCountMetadata metadata = count.getMetadata();

if (!metadata.dontFilterUser) {

long twitterUserId = metadata.getTwitterUserId();

int numResultsFromUser = 1;

if (twitterUserId != -1) {

Integer perUser = antiGamingMap.get(twitterUserId);

if (perUser != null) {

numResultsFromUser = perUser + 1;

filterOutUser = numResultsFromUser > maxHitsPerUser;

}

antiGamingMap.put(twitterUserId, numResultsFromUser);

}

}

}

// Filter facets those don't meet the basic criteria.

if (count.getSimpleCount() < minSimpleCount) {

if (DebugMessageBuilder.DEBUG\_VERBOSE <= debugMessageBuilder.getDebugLevel()) {

debugMessageBuilder.verbose2(

"[%d] FacetsResponseMerger EXCLUDED: simpleCount:%d < minSimpleCount:%d, %s",

i, count.getSimpleCount(), minSimpleCount, facetCountSummary(count));

}

continue;

}

if (count.getWeightedCount() < minWeightedCount) {

if (DebugMessageBuilder.DEBUG\_VERBOSE <= debugMessageBuilder.getDebugLevel()) {

debugMessageBuilder.verbose2(

"[%d] FacetsResponseMerger EXCLUDED: weightedCount:%d < minWeightedCount:%d, %s",

i, count.getWeightedCount(), minWeightedCount, facetCountSummary(count));

}

continue;

}

if (filterOutUser) {

if (DebugMessageBuilder.DEBUG\_VERBOSE <= debugMessageBuilder.getDebugLevel()) {

debugMessageBuilder.verbose2(

"[%d] FacetsResponseMerger EXCLUDED: antiGaming filterd user: %d: %s",

i, count.getMetadata().getTwitterUserId(), facetCountSummary(count));

}

continue;

}

if (count.getPenaltyCount() > maxPenaltyCount) {

if (DebugMessageBuilder.DEBUG\_VERBOSE <= debugMessageBuilder.getDebugLevel()) {

debugMessageBuilder.verbose2(

"[%d] FacetsResponseMerger EXCLUCED: penaltyCount:%.3f > maxPenaltyCount:%.3f, %s",

i, count.getPenaltyCount(), maxPenaltyCount, facetCountSummary(count));

}

continue;

}

if (((double) count.getPenaltyCount() / count.getSimpleCount()) > maxPenaltyCountRatio) {

if (DebugMessageBuilder.DEBUG\_VERBOSE <= debugMessageBuilder.getDebugLevel()) {

debugMessageBuilder.verbose2(

"[%d] FacetsResponseMerger EXCLUDED: penaltyCountRatio: %.3f > "

+ "maxPenaltyCountRatio:%.3f, %s",

i, (double) count.getPenaltyCount() / count.getSimpleCount(), maxPenaltyCountRatio,

facetCountSummary(count));

}

continue;

}

results.addToTopFacets(count);

collected++;

if (collected >= numResults) {

break;

}

}

}

private static int capFacetFieldWidth(int numResults) {

int ret = numResults;

if (numResults <= 0) {

// this in theory should not be allowed, but for now we issue the request with goodwill length

ret = 10; // default to 10 for future merge code to terminate correctly

}

if (numResults >= 100) {

ret = 100;

}

return ret;

}

private static String facetCountSummary(final ThriftFacetCount count) {

if (count.isSetMetadata()) {

return String.format("Label: %s (s:%d, w:%d, p:%d, score:%.2f, sid:%d (%s))",

count.getFacetLabel(), count.getSimpleCount(), count.getWeightedCount(),

count.getPenaltyCount(), count.getScore(), count.getMetadata().getStatusId(),

count.getMetadata().getStatusLanguage());

} else {

return String.format("Label: %s (s:%d, w:%d, p:%d, score:%.2f)", count.getFacetLabel(),

count.getSimpleCount(), count.getWeightedCount(), count.getPenaltyCount(),

count.getScore());

}

}

// Iterate through the backend responses and fill up the FacetFieldInfo map.

private void collectResponsesAndPopulateMap(

final Map<String, FacetsResultsUtils.FacetFieldInfo> facetFieldInfoMap,

final Set<Long> userIDWhitelist) {

// Next, iterate through the backend responses.

int i = 0;

for (EarlybirdResponse facetsResponse : accumulatedResponses.getSuccessResponses()) {

if (facetsResponse.isSetFacetResults()) {

LOG.debug("Facet response from earlybird {} is {} ", i, facetsResponse.getFacetResults());

i++;

ThriftFacetResults facetResults = facetsResponse.getFacetResults();

if (facetResults.isSetUserIDWhitelist()) {

userIDWhitelist.addAll(facetResults.getUserIDWhitelist());

}

FacetsResultsUtils.fillFacetFieldInfo(

facetResults, facetFieldInfoMap,

userIDWhitelist);

}

}

LOG.debug("Earlybird facet response total size {}", i);

}

}