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

import java.util.LinkedList;

import java.util.List;

import java.util.Set;

import com.google.common.base.Preconditions;

import com.google.common.collect.ImmutableSet;

import org.apache.lucene.search.Query;

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

import com.twitter.search.common.search.TerminationTracker;

import com.twitter.search.common.util.text.NormalizerHelper;

import com.twitter.search.common.util.url.URLUtils;

import com.twitter.search.earlybird.common.config.EarlybirdConfig;

import com.twitter.search.earlybird.search.SearchRequestInfo;

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

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

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

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

public class TermStatisticsRequestInfo extends SearchRequestInfo {

private static final Set<String> FACET\_URL\_FIELDS\_TO\_NORMALIZE = new ImmutableSet.Builder()

.add(EarlybirdFieldConstant.IMAGES\_FACET)

.add(EarlybirdFieldConstant.VIDEOS\_FACET)

.add(EarlybirdFieldConstant.NEWS\_FACET)

.build();

protected final List<ThriftTermRequest> termRequests;

protected final ThriftHistogramSettings histogramSettings;

/\*\*

\* Creates a new TermStatisticsRequestInfo instance using the provided query.

\*/

public TermStatisticsRequestInfo(ThriftSearchQuery searchQuery,

Query luceneQuery,

ThriftTermStatisticsRequest termStatsRequest,

TerminationTracker terminationTracker) {

super(searchQuery, luceneQuery, terminationTracker);

this.termRequests = termStatsRequest.isSetTermRequests()

? termStatsRequest.getTermRequests() : new LinkedList<>();

this.histogramSettings = termStatsRequest.getHistogramSettings();

if (termStatsRequest.isIncludeGlobalCounts()) {

// Add an empty request to indicate we need a global count across all fields.

termRequests.add(new ThriftTermRequest().setFieldName("").setTerm(""));

}

// We only normalize TEXT terms and urls. All other terms, e.g. topics (named entities) are

// not normalized. Here the assumption is that the caller passes the exact terms back that

// the facet API returned

for (ThriftTermRequest termReq : termRequests) {

if (termReq.getTerm().isEmpty()) {

continue; // the special catch-all term.

}

if (!termReq.isSetFieldName()

|| termReq.getFieldName().equals(EarlybirdFieldConstant.TEXT\_FIELD.getFieldName())) {

// normalize the TEXT term as it's normalized during ingestion

termReq.setTerm(NormalizerHelper.normalizeWithUnknownLocale(

termReq.getTerm(), EarlybirdConfig.getPenguinVersion()));

} else if (FACET\_URL\_FIELDS\_TO\_NORMALIZE.contains(termReq.getFieldName())) {

// remove the trailing slash from the URL path. This operation is idempotent,

// so either a spiderduck URL or a facet URL can be used here. The latter would just

// be normalized twice, which is fine.

termReq.setTerm(URLUtils.normalizePath(termReq.getTerm()));

}

}

}

@Override

protected int calculateMaxHitsToProcess(ThriftSearchQuery searchQuery) {

Preconditions.checkNotNull(searchQuery.getCollectorParams());

if (!searchQuery.getCollectorParams().isSetTerminationParams()

|| !searchQuery.getCollectorParams().getTerminationParams().isSetMaxHitsToProcess()) {

// Override the default value to all hits.

return Integer.MAX\_VALUE;

} else {

return super.calculateMaxHitsToProcess(searchQuery);

}

}

public final List<ThriftTermRequest> getTermRequests() {

return this.termRequests;

}

public final ThriftHistogramSettings getHistogramSettings() {

return this.histogramSettings;

}

public final boolean isReturnHistogram() {

return this.histogramSettings != null;

}

}