package com.twitter.search.earlybird\_root;

import java.util.List;

import java.util.Map;

import javax.inject.Inject;

import javax.inject.Named;

import com.google.common.annotations.VisibleForTesting;

import com.google.common.base.Predicate;

import com.google.common.collect.Maps;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.twitter.finagle.Service;

import com.twitter.finagle.SimpleFilter;

import com.twitter.search.common.decider.SearchDecider;

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

import com.twitter.search.common.root.SearchRootModule;

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

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

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

import com.twitter.search.queryparser.query.Query;

import com.twitter.search.queryparser.query.QueryParserException;

import com.twitter.search.queryparser.query.Term;

import com.twitter.search.queryparser.query.annotation.Annotation;

import com.twitter.search.queryparser.rewriter.PredicateQueryNodeDropper;

import com.twitter.search.queryparser.visitors.TermExtractorVisitor;

import com.twitter.util.Future;

/\*\*

\* Filter that rewrites the serialized query on EarlybirdRequest.

\* As of now, this filter performs the following rewrites:

\* - Drop ":v annotated variants based on decider, if the query has enough term nodes.

\*/

public class EarlybirdQueryRewriteFilter extends

SimpleFilter<EarlybirdRequestContext, EarlybirdResponse> {

private static final Logger LOG =

LoggerFactory.getLogger(EarlybirdQueryRewriteFilter.class);

private static final String DROP\_PHRASE\_VARIANT\_FROM\_QUERY\_DECIDER\_KEY\_PATTERN =

"drop\_variants\_from\_%s\_%s\_queries";

// only drop variants from queries with more than this number of terms.

private static final String MIN\_TERM\_COUNT\_FOR\_VARIANT\_DROPPING\_DECIDER\_KEY\_PATTERN =

"drop\_variants\_from\_%s\_%s\_queries\_term\_count\_threshold";

private static final SearchCounter QUERY\_PARSER\_FAILURE\_COUNT =

SearchCounter.export("query\_rewrite\_filter\_query\_parser\_failure\_count");

// We currently add variants only to RECENCY and RELEVANCE requests, but it doesn't hurt to export

// stats for all request types.

@VisibleForTesting

static final Map<EarlybirdRequestType, SearchCounter> DROP\_VARIANTS\_QUERY\_COUNTS =

Maps.newEnumMap(EarlybirdRequestType.class);

static {

for (EarlybirdRequestType requestType : EarlybirdRequestType.values()) {

DROP\_VARIANTS\_QUERY\_COUNTS.put(

requestType,

SearchCounter.export(String.format("drop\_%s\_variants\_query\_count",

requestType.getNormalizedName())));

}

}

private static final Predicate<Query> DROP\_VARIANTS\_PREDICATE =

q -> q.hasAnnotationType(Annotation.Type.VARIANT);

private static final PredicateQueryNodeDropper DROP\_VARIANTS\_VISITOR =

new PredicateQueryNodeDropper(DROP\_VARIANTS\_PREDICATE);

private final SearchDecider decider;

private final String normalizedSearchRootName;

@Inject

public EarlybirdQueryRewriteFilter(

SearchDecider decider,

@Named(SearchRootModule.NAMED\_NORMALIZED\_SEARCH\_ROOT\_NAME) String normalizedSearchRootName) {

this.decider = decider;

this.normalizedSearchRootName = normalizedSearchRootName;

}

@Override

public Future<EarlybirdResponse> apply(

EarlybirdRequestContext requestContext,

Service<EarlybirdRequestContext, EarlybirdResponse> service) {

Query query = requestContext.getParsedQuery();

// If there's no serialized query, no rewrite is necessary.

if (query == null) {

return service.apply(requestContext);

} else {

try {

Query variantsRemoved = maybeRemoveVariants(requestContext, query);

if (query == variantsRemoved) {

return service.apply(requestContext);

} else {

EarlybirdRequestContext clonedRequestContext =

EarlybirdRequestContext.copyRequestContext(requestContext, variantsRemoved);

return service.apply(clonedRequestContext);

}

} catch (QueryParserException e) {

// It is not clear here that the QueryParserException is the client's fault, or our fault.

// At this point it is most likely not the client's since we have a legitimate parsed Query

// from the client's request, and it's the rewriting that failed.

// In this case we choose to send the query as is (without the rewrite), instead of

// failing the entire request.

QUERY\_PARSER\_FAILURE\_COUNT.increment();

LOG.warn("Failed to rewrite serialized query: " + query.serialize(), e);

return service.apply(requestContext);

}

}

}

private Query maybeRemoveVariants(EarlybirdRequestContext requestContext, Query query)

throws QueryParserException {

if (shouldDropVariants(requestContext, query)) {

Query rewrittenQuery = DROP\_VARIANTS\_VISITOR.apply(query);

if (!query.equals(rewrittenQuery)) {

DROP\_VARIANTS\_QUERY\_COUNTS.get(requestContext.getEarlybirdRequestType()).increment();

return rewrittenQuery;

}

}

return query;

}

private boolean shouldDropVariants(EarlybirdRequestContext requestContext, Query query)

throws QueryParserException {

TermExtractorVisitor termExtractorVisitor = new TermExtractorVisitor(false);

List<Term> terms = query.accept(termExtractorVisitor);

EarlybirdRequestType requestType = requestContext.getEarlybirdRequestType();

boolean shouldDropVariants = decider.isAvailable(getDropPhaseVariantDeciderKey(requestType));

return terms != null

&& terms.size() >= decider.getAvailability(

getMinTermCountForVariantDroppingDeciderKey(requestType))

&& shouldDropVariants;

}

private String getDropPhaseVariantDeciderKey(EarlybirdRequestType requestType) {

return String.format(DROP\_PHRASE\_VARIANT\_FROM\_QUERY\_DECIDER\_KEY\_PATTERN,

normalizedSearchRootName,

requestType.getNormalizedName());

}

private String getMinTermCountForVariantDroppingDeciderKey(EarlybirdRequestType requestType) {

return String.format(MIN\_TERM\_COUNT\_FOR\_VARIANT\_DROPPING\_DECIDER\_KEY\_PATTERN,

normalizedSearchRootName,

requestType.getNormalizedName());

}

}