package com.twitter.search.earlybird.search.relevance.collectors;

import java.io.IOException;

import com.google.common.base.Preconditions;

import com.twitter.common.util.Clock;

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

import com.twitter.search.common.schema.earlybird.EarlybirdCluster;

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

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

import com.twitter.search.earlybird.common.userupdates.UserTable;

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

import com.twitter.search.earlybird.search.relevance.RelevanceSearchRequestInfo;

import com.twitter.search.earlybird.search.relevance.RelevanceSearchResults;

import com.twitter.search.earlybird.search.relevance.scoring.ScoringFunction;

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

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

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

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

/\*\*

\* AbstractRelevanceCollector is a results collector that collects RelevanceHit results

\* which include more detailed information than a normal Hit.

\*/

public abstract class AbstractRelevanceCollector

extends AbstractResultsCollector<RelevanceSearchRequestInfo, RelevanceSearchResults> {

protected final ScoringFunction scoringFunction;

private final ThriftSearchResultsRelevanceStats relevanceStats;

private final EarlybirdCluster cluster;

private final UserTable userTable;

// Per-language result counts.

private final LanguageHistogram languageHistogram = new LanguageHistogram();

// Accumulated time spend on relevance scoring across all collected hits, including batch scoring.

private long scoringTimeNanos = 0;

public AbstractRelevanceCollector(

ImmutableSchemaInterface schema,

RelevanceSearchRequestInfo searchRequestInfo,

ScoringFunction scoringFunction,

EarlybirdSearcherStats searcherStats,

EarlybirdCluster cluster,

UserTable userTable,

Clock clock,

int requestDebugMode) {

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

this.scoringFunction = scoringFunction;

this.relevanceStats = new ThriftSearchResultsRelevanceStats();

this.cluster = cluster;

this.userTable = userTable;

}

/\*\*

\* Subclasses must implement this method to actually collect a scored relevance hit.

\*/

protected abstract void doCollectWithScore(long tweetID, float score) throws IOException;

@Override

public final void startSegment() throws IOException {

scoringFunction.setNextReader(currTwitterReader);

ThriftSearchResultMetadataOptions options =

searchRequestInfo.getSearchQuery().getResultMetadataOptions();

featuresRequested = options != null && options.isReturnSearchResultFeatures();

}

@Override

protected final void doCollect(long tweetID) throws IOException {

final long scoringStartNanos = getClock().nowNanos();

float luceneSore = scorer.score();

final float score = scoringFunction.score(curDocId, luceneSore);

final long scoringEndNanos = getClock().nowNanos();

addToOverallScoringTimeNanos(scoringStartNanos, scoringEndNanos);

scoringFunction.updateRelevanceStats(relevanceStats);

updateHitCounts(tweetID);

doCollectWithScore(tweetID, score);

}

protected final void addToOverallScoringTimeNanos(long scoringStartNanos, long scoringEndNanos) {

scoringTimeNanos += scoringEndNanos - scoringStartNanos;

}

protected final ThriftSearchResultMetadata collectMetadata() throws IOException {

ThriftSearchResultMetadataOptions options =

searchRequestInfo.getSearchQuery().getResultMetadataOptions();

Preconditions.checkNotNull(options);

ThriftSearchResultMetadata metadata =

Preconditions.checkNotNull(scoringFunction.getResultMetadata(options));

if (metadata.isSetLanguage()) {

languageHistogram.increment(metadata.getLanguage().getValue());

}

// Some additional metadata which is not provided by the scoring function, but

// by accessing the reader directly.

if (currTwitterReader != null) {

fillResultGeoLocation(metadata);

if (searchRequestInfo.isCollectConversationId()) {

long conversationId =

documentFeatures.getFeatureValue(EarlybirdFieldConstant.CONVERSATION\_ID\_CSF);

if (conversationId != 0) {

ensureExtraMetadataIsSet(metadata);

metadata.getExtraMetadata().setConversationId(conversationId);

}

}

}

// Check and collect hit attribution data, if it's available.

fillHitAttributionMetadata(metadata);

long fromUserId = documentFeatures.getFeatureValue(EarlybirdFieldConstant.FROM\_USER\_ID\_CSF);

if (searchRequestInfo.isGetFromUserId()) {

metadata.setFromUserId(fromUserId);

}

collectExclusiveConversationAuthorId(metadata);

collectFacets(metadata);

collectFeatures(metadata);

collectIsProtected(metadata, cluster, userTable);

return metadata;

}

protected final ThriftSearchResultsRelevanceStats getRelevanceStats() {

return relevanceStats;

}

public final LanguageHistogram getLanguageHistogram() {

return languageHistogram;

}

@Override

protected final RelevanceSearchResults doGetResults() throws IOException {

final RelevanceSearchResults results = doGetRelevanceResults();

results.setScoringTimeNanos(scoringTimeNanos);

return results;

}

/\*\*

\* For subclasses to process and aggregate collected hits.

\*/

protected abstract RelevanceSearchResults doGetRelevanceResults() throws IOException;

}