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

import com.twitter.search.common.relevance.utils.ResultComparators;

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

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

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

/\*\*

\* RelevanceMergeCollector class extends (@link RecencyMergeCollector} to do k-way merge of

\* earlybird responses, but sorted by relevance score.

\*

\* Note that this is a superset of functionality found in

\* {@link com.twitter.search.blender.services.earlybird.relevance.RelevanceCollector}

\* If you make changes here, evaluate if they should be made in RelevanceCollector as well.

\*/

public class RelevanceMergeCollector extends RecencyMergeCollector {

public RelevanceMergeCollector(int numResponses) {

super(numResponses, ResultComparators.SCORE\_COMPARATOR);

}

@Override

protected void collectStats(EarlybirdResponse response) {

super.collectStats(response);

if (!response.getSearchResults().isSetRelevanceStats()) {

return;

}

if (!finalResults.isSetRelevanceStats()) {

finalResults.setRelevanceStats(new ThriftSearchResultsRelevanceStats());

}

ThriftSearchResultsRelevanceStats base = finalResults.getRelevanceStats();

ThriftSearchResultsRelevanceStats delta = response.getSearchResults().getRelevanceStats();

ThriftSearchResultsRelevanceStatsUtil.addRelevanceStats(base, delta);

}

}