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

import java.util.Comparator;

import javax.annotation.Nullable;

import com.google.common.base.Preconditions;

import com.twitter.common\_internal.collections.RandomAccessPriorityQueue;

import com.twitter.search.common.relevance.features.TweetIntegerShingleSignature;

import com.twitter.search.earlybird.search.Hit;

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

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

public class RelevanceHit extends Hit

implements RandomAccessPriorityQueue.SignatureProvider<TweetIntegerShingleSignature> {

@Nullable

private TweetIntegerShingleSignature signature;

public RelevanceHit() {

super(Long.MAX\_VALUE, Long.MAX\_VALUE);

}

public RelevanceHit(long timeSliceID, long statusID,

TweetIntegerShingleSignature signature,

ThriftSearchResultMetadata metadata) {

super(timeSliceID, statusID);

update(timeSliceID, statusID, signature, metadata);

}

/\*\*

\* Updates the data for this relevance hit.

\*

\* @param timeSliceID The timeslice ID of the segment that the segment came from.

\* @param statusID The hit's tweet ID.

\* @param tweetSignature The tweet signature generated for this hit.

\* @param metadata The metadata associated with this hit.

\*/

public void update(long timeSliceID, long statusID, TweetIntegerShingleSignature tweetSignature,

ThriftSearchResultMetadata metadata) {

this.statusID = statusID;

this.timeSliceID = timeSliceID;

this.metadata = Preconditions.checkNotNull(metadata);

this.signature = Preconditions.checkNotNull(tweetSignature);

}

/\*\*

\* Returns the computed score for this hit.

\*/

public float getScore() {

if (metadata != null) {

return (float) metadata.getScore();

} else {

return ScoringFunction.SKIP\_HIT;

}

}

// We want the score as a double (and not cast to a float) for COMPARATOR\_BY\_SCORE and

// PQ\_COMPARATOR\_BY\_SCORE so that the results returned from Earlybirds will be sorted based on the

// scores in the ThriftSearchResultMetadata objects (and will not lose precision by being cast to

// floats). Thus, the sorted order on Earlybirds and Earlybird Roots will be consistent.

private double getScoreDouble() {

if (metadata != null) {

return metadata.getScore();

} else {

return (double) ScoringFunction.SKIP\_HIT;

}

}

@Override @Nullable

public TweetIntegerShingleSignature getSignature() {

return signature;

}

@Override

public String toString() {

return "RelevanceHit[tweetID=" + statusID + ",timeSliceID=" + timeSliceID

+ ",score=" + (metadata == null ? "null" : metadata.getScore())

+ ",signature=" + (signature == null ? "null" : signature) + "]";

}

public static final Comparator<RelevanceHit> COMPARATOR\_BY\_SCORE =

(d1, d2) -> {

// if two docs have the same score, then the first one (most recent) wins

if (d1.getScore() == d2.getScore()) {

return Long.compare(d2.getStatusID(), d1.getStatusID());

}

return Double.compare(d2.getScoreDouble(), d1.getScoreDouble());

};

public static final Comparator<RelevanceHit> PQ\_COMPARATOR\_BY\_SCORE =

(d1, d2) -> {

// Reverse the order

return COMPARATOR\_BY\_SCORE.compare(d2, d1);

};

@Override

public void clear() {

timeSliceID = Long.MAX\_VALUE;

statusID = Long.MAX\_VALUE;

metadata = null;

signature = null;

}

}