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

import org.apache.lucene.search.Explanation;

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

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

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

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

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

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

/\*\*

\* A dummy scoring function for test, the score is always tweetId/10000.0

\* Since score\_filter: operator requires all score to be between [0, 1], if you want to use this

\* with it, don't use any tweet id larger than 10000 in your test.

\*/

public class TestScoringFunction extends ScoringFunction {

private ThriftSearchResultMetadata metadata = null;

private float score;

public TestScoringFunction(ImmutableSchemaInterface schema) {

super(schema);

}

@Override

protected float score(float luceneQueryScore) {

long tweetId = tweetIDMapper.getTweetID(getCurrentDocID());

this.score = (float) (tweetId / 10000.0);

System.out.println(String.format("score for tweet %10d is %6.3f", tweetId, score));

return this.score;

}

@Override

protected Explanation doExplain(float luceneScore) {

return null;

}

@Override

public ThriftSearchResultMetadata getResultMetadata(ThriftSearchResultMetadataOptions options) {

if (metadata == null) {

metadata = new ThriftSearchResultMetadata()

.setResultType(ThriftSearchResultType.RELEVANCE)

.setPenguinVersion(EarlybirdConfig.getPenguinVersionByte());

metadata.setScore(score);

}

return metadata;

}

@Override

public void updateRelevanceStats(ThriftSearchResultsRelevanceStats relevanceStats) {

}

}