package com.twitter.home\_mixer.product.for\_you.query\_transformer

import com.twitter.conversions.DurationOps.richDurationFromInt

import com.twitter.finagle.thrift.ClientId

import com.twitter.finagle.tracing.Trace

import com.twitter.product\_mixer.component\_library.feature\_hydrator.query.social\_graph.PreviewCreatorsFeature

import com.twitter.product\_mixer.core.functional\_component.transformer.CandidatePipelineQueryTransformer

import com.twitter.product\_mixer.core.pipeline.PipelineQuery

import com.twitter.search.common.ranking.{thriftscala => scr}

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

import com.twitter.search.earlybird.{thriftscala => t}

import com.twitter.search.queryparser.query.Conjunction

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

import com.twitter.search.queryparser.query.search.SearchOperator

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class TweetPreviewsQueryTransformer @Inject() (clientId: ClientId)

extends CandidatePipelineQueryTransformer[PipelineQuery, t.EarlybirdRequest] {

private val MaxPreviewTweets = 200

private val EarlybirdRelevanceTensorflowModel = "timelines\_rectweet\_replica"

private val SinceDuration = 7.days

val MetadataOptions = t.ThriftSearchResultMetadataOptions(

getReferencedTweetAuthorId = true,

getFromUserId = true

)

override def transform(query: PipelineQuery): t.EarlybirdRequest = {

val candidatePreviewCreatorIds =

query.features.map(\_.get(PreviewCreatorsFeature)).getOrElse(Seq.empty)

val searchQuery = new Conjunction(

// Include subscriber only (aka exclusive) Tweets

new SearchOperator.Builder()

.setType(SearchOperator.Type.FILTER)

.addOperand(EarlybirdFieldConstant.EXCLUSIVE\_FILTER\_TERM)

.build(),

// Include only original Tweets

new SearchOperator.Builder()

.setType(SearchOperator.Type.FILTER)

.addOperand(EarlybirdFieldConstant.NATIVE\_RETWEETS\_FILTER\_TERM)

.setOccur(Query.Occur.MUST\_NOT)

.build(),

new SearchOperator.Builder()

.setType(SearchOperator.Type.FILTER)

.addOperand(EarlybirdFieldConstant.REPLIES\_FILTER\_TERM)

.setOccur(Query.Occur.MUST\_NOT)

.build(),

new SearchOperator.Builder()

.setType(SearchOperator.Type.FILTER)

.addOperand(EarlybirdFieldConstant.QUOTE\_FILTER\_TERM)

.setOccur(Query.Occur.MUST\_NOT)

.build(),

new SearchOperator(SearchOperator.Type.SINCE\_TIME, SinceDuration.ago.inSeconds.toString)

)

t.EarlybirdRequest(

searchQuery = t.ThriftSearchQuery(

serializedQuery = Some(searchQuery.serialize),

fromUserIDFilter64 = Some(candidatePreviewCreatorIds),

numResults = MaxPreviewTweets,

rankingMode = t.ThriftSearchRankingMode.Relevance,

relevanceOptions = Some(

t.ThriftSearchRelevanceOptions(

filterDups = true,

keepDupWithHigherScore = true,

proximityScoring = true,

maxConsecutiveSameUser = Some(5),

rankingParams = Some(

scr.ThriftRankingParams(

`type` = Some(scr.ThriftScoringFunctionType.TensorflowBased),

selectedTensorflowModel = Some(EarlybirdRelevanceTensorflowModel),

minScore = -1.0e100,

applyBoosts = false,

)

),

),

),

resultMetadataOptions = Some(MetadataOptions),

searcherId = query.getOptionalUserId,

),

getOlderResults = Some(true), // needed for archive access to older tweets

clientRequestID = Some(s"${Trace.id.traceId}"),

followedUserIds = Some(candidatePreviewCreatorIds.toSeq),

numResultsToReturnAtRoot = Some(MaxPreviewTweets),

clientId = Some(clientId.name),

)

}

}