package com.twitter.home\_mixer.product.scored\_tweets.query\_transformer.earlybird

import com.twitter.home\_mixer.model.HomeFeatures.RealGraphInNetworkScoresFeature

import com.twitter.home\_mixer.model.request.HasDeviceContext

import com.twitter.home\_mixer.util.CachedScoredTweetsHelper

import com.twitter.home\_mixer.util.earlybird.EarlybirdRequestUtil

import com.twitter.product\_mixer.core.model.common.identifier.CandidatePipelineIdentifier

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

import com.twitter.product\_mixer.core.quality\_factor.HasQualityFactorStatus

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

import com.twitter.timelines.clients.relevance\_search.SearchClient.TweetTypes

import com.twitter.timelines.common.model.TweetKindOption

import com.twitter.timelines.util.SnowflakeSortIndexHelper

import com.twitter.util.Duration

import com.twitter.util.Time

trait EarlybirdQueryTransformer[

Query <: PipelineQuery with HasQualityFactorStatus with HasDeviceContext] {

def candidatePipelineIdentifier: CandidatePipelineIdentifier

def clientId: Option[String] = None

def maxTweetsToFetch: Int = 100

def tweetKindOptions: TweetKindOption.ValueSet

def tensorflowModel: Option[String] = None

private val EarlybirdMaxExcludedTweets = 1500

def buildEarlybirdQuery(

query: Query,

sinceDuration: Duration,

followedUserIds: Set[Long] = Set.empty

): eb.EarlybirdRequest = {

val sinceTime: Time = sinceDuration.ago

val untilTime: Time = Time.now

val fromTweetIdExclusive = SnowflakeSortIndexHelper.timestampToFakeId(sinceTime)

val toTweetIdExclusive = SnowflakeSortIndexHelper.timestampToFakeId(untilTime)

val excludedTweetIds = query.features.map { featureMap =>

CachedScoredTweetsHelper.tweetImpressionsAndCachedScoredTweetsInRange(

featureMap,

candidatePipelineIdentifier,

EarlybirdMaxExcludedTweets,

sinceTime,

untilTime)

}

val maxCount =

(query.getQualityFactorCurrentValue(candidatePipelineIdentifier) \* maxTweetsToFetch).toInt

val authorScoreMap = query.features

.map(\_.getOrElse(RealGraphInNetworkScoresFeature, Map.empty[Long, Double]))

.getOrElse(Map.empty)

EarlybirdRequestUtil.getTweetsRequest(

userId = Some(query.getRequiredUserId),

clientId = clientId,

skipVeryRecentTweets = true,

followedUserIds = followedUserIds,

retweetsMutedUserIds = Set.empty,

beforeTweetIdExclusive = Some(toTweetIdExclusive),

afterTweetIdExclusive = Some(fromTweetIdExclusive),

excludedTweetIds = excludedTweetIds.map(\_.toSet),

maxCount = maxCount,

tweetTypes = TweetTypes.fromTweetKindOption(tweetKindOptions),

authorScoreMap = Some(authorScoreMap),

tensorflowModel = tensorflowModel

)

}

}