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

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

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

import com.twitter.home\_mixer.product.scored\_tweets.query\_transformer.TimelineRankerQueryTransformer.\_

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.timelineranker.{model => tlr}

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

import com.twitter.timelines.earlybird.common.options.EarlybirdOptions

import com.twitter.timelines.earlybird.common.options.EarlybirdScoringModelConfig

import com.twitter.timelines.earlybird.common.utils.SearchOperator

import com.twitter.timelines.model.UserId

import com.twitter.timelines.model.candidate.CandidateTweetSourceId

import com.twitter.timelines.util.SnowflakeSortIndexHelper

import com.twitter.util.Duration

import com.twitter.util.Time

object TimelineRankerQueryTransformer {

/\*\*

\* Specifies the maximum number of excluded tweet ids to include in the search index query.

\* Earlybird's named multi term disjunction map feature supports up to 1500 tweet ids.

\*/

private val EarlybirdMaxExcludedTweets = 1500

/\*\*

\* Maximum number of query hits each earlybird shard is allowed to accumulate before

\* early-terminating the query and reducing the hits to MaxNumEarlybirdResults.

\*/

private val EarlybirdMaxHits = 1000

/\*\*

\* Maximum number of results TLR should retrieve from each earlybird shard.

\*/

private val EarlybirdMaxResults = 300

}

trait TimelineRankerQueryTransformer[

Query <: PipelineQuery with HasQualityFactorStatus with HasDeviceContext] {

def maxTweetsToFetch: Int

def options: TweetKindOption.ValueSet = TweetKindOption.Default

def candidateTweetSourceId: CandidateTweetSourceId.Value

def utegLikedByTweetsOptions(query: Query): Option[tlr.UtegLikedByTweetsOptions] = None

def seedAuthorIds(query: Query): Option[Seq[Long]] = None

def candidatePipelineIdentifier: CandidatePipelineIdentifier

def earlybirdModels: Seq[EarlybirdScoringModelConfig] =

EarlybirdRequestUtil.EarlybirdScoringModels.UnifiedEngagementProd

def getTensorflowModel(query: Query): Option[String] = None

def buildTimelineRankerQuery(query: Query, sinceDuration: Duration): tlr.RecapQuery = {

val sinceTime: Time = sinceDuration.ago

val untilTime: Time = Time.now

val fromTweetIdExclusive = SnowflakeSortIndexHelper.timestampToFakeId(sinceTime)

val toTweetIdExclusive = SnowflakeSortIndexHelper.timestampToFakeId(untilTime)

val range = tlr.TweetIdRange(Some(fromTweetIdExclusive), Some(toTweetIdExclusive))

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[UserId, Double]))

.getOrElse(Map.empty)

val deviceContext =

query.deviceContext.map(\_.toTimelineServiceDeviceContext(query.clientContext))

val tensorflowModel = getTensorflowModel(query)

val earlyBirdOptions = EarlybirdOptions(

maxNumHitsPerShard = EarlybirdMaxHits,

maxNumResultsPerShard = EarlybirdMaxResults,

models = earlybirdModels,

authorScoreMap = authorScoreMap,

skipVeryRecentTweets = true,

tensorflowModel = tensorflowModel

)

tlr.RecapQuery(

userId = query.getRequiredUserId,

maxCount = Some(maxCount),

range = Some(range),

options = options,

searchOperator = SearchOperator.Exclude,

earlybirdOptions = Some(earlyBirdOptions),

deviceContext = deviceContext,

authorIds = seedAuthorIds(query),

excludedTweetIds = excludedTweetIds,

utegLikedByTweetsOptions = utegLikedByTweetsOptions(query),

searchClientSubId = None,

candidateTweetSourceId = Some(candidateTweetSourceId),

hydratesContentFeatures = Some(false)

)

}

}