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

import com.twitter.conversions.DurationOps.\_

import com.twitter.core\_workflows.user\_model.{thriftscala => um}

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

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

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

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

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

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.common.model.TweetKindOption

object EarlybirdInNetworkQueryTransformer {

private val DefaultSinceDuration = 24.hours

private val ExpandedSinceDuration = 48.hours

private val MaxTweetsToFetch = 600

private val TensorflowModel = Some("timelines\_recap\_replica")

private val TweetKindOptions: TweetKindOption.ValueSet = TweetKindOption(

includeReplies = true,

includeRetweets = true,

includeOriginalTweetsAndQuotes = true,

includeExtendedReplies = true

)

private val UserStatesForExtendedSinceDuration: Set[um.UserState] = Set(

um.UserState.Light,

um.UserState.MediumNonTweeter,

um.UserState.MediumTweeter,

um.UserState.NearZero,

um.UserState.New,

um.UserState.VeryLight

)

}

case class EarlybirdInNetworkQueryTransformer[

Query <: PipelineQuery with HasQualityFactorStatus with HasDeviceContext

](

candidatePipelineIdentifier: CandidatePipelineIdentifier,

override val clientId: Option[String])

extends CandidatePipelineQueryTransformer[Query, eb.EarlybirdRequest]

with EarlybirdQueryTransformer[Query] {

override val tweetKindOptions: TweetKindOption.ValueSet = TweetKindOptions

override val maxTweetsToFetch: Int = MaxTweetsToFetch

override val tensorflowModel: Option[String] = TensorflowModel

override def transform(query: Query): eb.EarlybirdRequest = {

val userState = query.features.get.getOrElse(UserStateFeature, None)

val sinceDuration =

if (userState.exists(UserStatesForExtendedSinceDuration.contains)) ExpandedSinceDuration

else DefaultSinceDuration

val followedUserIds =

query.features

.map(

\_.getOrElse(

SGSFollowedUsersFeature,

Seq.empty)).toSeq.flatten.toSet + query.getRequiredUserId

buildEarlybirdQuery(query, sinceDuration, followedUserIds)

}

}