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

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.param.ScoredTweetsParam

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

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.timelineranker.{thriftscala => t}

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

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

object TimelineRankerInNetworkQueryTransformer {

private val DefaultSinceDuration = 24.hours

private val ExpandedSinceDuration = 48.hours

private val MaxTweetsToFetch = 600

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 TimelineRankerInNetworkQueryTransformer[

Query <: PipelineQuery with HasQualityFactorStatus with HasDeviceContext

](

override val candidatePipelineIdentifier: CandidatePipelineIdentifier,

override val maxTweetsToFetch: Int = MaxTweetsToFetch)

extends CandidatePipelineQueryTransformer[Query, t.RecapQuery]

with TimelineRankerQueryTransformer[Query] {

override val candidateTweetSourceId = CandidateTweetSourceId.RecycledTweet

override val options = tweetKindOptions

override def getTensorflowModel(query: Query): Option[String] = {

Some(query.params(ScoredTweetsParam.EarlybirdTensorflowModel.InNetworkParam))

}

override def transform(input: Query): t.RecapQuery = {

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

val sinceDuration =

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

else DefaultSinceDuration

buildTimelineRankerQuery(input, sinceDuration).toThriftRecapQuery

}

}