package com.twitter.usersignalservice.signals

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.twistly.common.UserId

import com.twitter.twistly.thriftscala.UserRecentVideoViewTweets

import com.twitter.twistly.thriftscala.VideoViewEngagementType

import com.twitter.usersignalservice.base.Query

import com.twitter.usersignalservice.thriftscala.Signal

import com.twitter.util.Future

import com.twitter.util.Timer

import com.twitter.twistly.thriftscala.RecentVideoViewTweet

import com.twitter.usersignalservice.thriftscala.SignalType

import com.twitter.simclusters\_v2.thriftscala.InternalId

import com.twitter.strato.client.Client

import com.twitter.strato.data.Conv

import com.twitter.strato.thrift.ScroogeConv

import com.twitter.usersignalservice.base.StratoSignalFetcher

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

case class VideoTweetsQualityViewFetcher @Inject() (

stratoClient: Client,

timer: Timer,

stats: StatsReceiver)

extends StratoSignalFetcher[

(UserId, VideoViewEngagementType),

Unit,

UserRecentVideoViewTweets

] {

import VideoTweetsQualityViewFetcher.\_

override type RawSignalType = RecentVideoViewTweet

override def name: String = this.getClass.getCanonicalName

override def statsReceiver: StatsReceiver = stats.scope(name)

override val stratoColumnPath: String = StratoColumn

override val stratoView: Unit = None

override protected val keyConv: Conv[(UserId, VideoViewEngagementType)] = Conv.ofType

override protected val viewConv: Conv[Unit] = Conv.ofType

override protected val valueConv: Conv[UserRecentVideoViewTweets] =

ScroogeConv.fromStruct[UserRecentVideoViewTweets]

override protected def toStratoKey(userId: UserId): (UserId, VideoViewEngagementType) =

(userId, VideoViewEngagementType.VideoQualityView)

override protected def toRawSignals(

stratoValue: UserRecentVideoViewTweets

): Seq[RecentVideoViewTweet] = stratoValue.recentEngagedTweets

override def process(

query: Query,

rawSignals: Future[Option[Seq[RecentVideoViewTweet]]]

): Future[Option[Seq[Signal]]] = {

rawSignals.map {

\_.map {

\_.filter(videoView =>

!videoView.isPromotedTweet && videoView.videoDurationSeconds >= MinVideoDurationSeconds)

.map { rawSignal =>

Signal(

SignalType.VideoView90dQualityV1,

rawSignal.engagedAt,

Some(InternalId.TweetId(rawSignal.tweetId)))

}.take(query.maxResults.getOrElse(Int.MaxValue))

}

}

}

}

object VideoTweetsQualityViewFetcher {

private val StratoColumn = "recommendations/twistly/userRecentVideoViewTweetEngagements"

private val MinVideoDurationSeconds = 10

}