package com.twitter.home\_mixer.product.scored\_tweets.feature\_hydrator

import com.twitter.finagle.stats.StatsReceiver

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

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

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

import com.twitter.product\_mixer.component\_library.model.candidate.TweetCandidate

import com.twitter.product\_mixer.core.feature.Feature

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMap

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMapBuilder

import com.twitter.product\_mixer.core.functional\_component.feature\_hydrator.BulkCandidateFeatureHydrator

import com.twitter.product\_mixer.core.model.common.CandidateWithFeatures

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

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

import com.twitter.product\_mixer.core.util.OffloadFuturePools

import com.twitter.stitch.Stitch

import com.twitter.strato.generated.client.audiencerewards.audienceRewardsService.GetSuperFollowEligibilityOnUserClientColumn

import com.twitter.util.Throw

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class AuthorIsCreatorFeatureHydrator @Inject() (

getSuperFollowEligibilityOnUserClientColumn: GetSuperFollowEligibilityOnUserClientColumn,

statsReceiver: StatsReceiver)

extends BulkCandidateFeatureHydrator[PipelineQuery, TweetCandidate] {

override val identifier: FeatureHydratorIdentifier =

FeatureHydratorIdentifier("AuthorIsCreator")

override val features: Set[Feature[\_, \_]] =

Set(AuthorIsCreatorFeature)

private val scopedStatsReceiver = statsReceiver.scope(getClass.getSimpleName)

private val keyFoundCounter = scopedStatsReceiver.counter("key/found")

private val keyFailureCounter = scopedStatsReceiver.counter("key/failure")

private val MissingKeyFeatureMap = FeatureMapBuilder()

.add(AuthorIsCreatorFeature, Throw(MissingKeyException))

.build()

private val DefaultFeatureMap = FeatureMapBuilder()

.add(AuthorIsCreatorFeature, false)

.build()

override def apply(

query: PipelineQuery,

candidates: Seq[CandidateWithFeatures[TweetCandidate]]

): Stitch[Seq[FeatureMap]] = {

OffloadFuturePools.offloadStitch {

val authorIds = candidates.flatMap(\_.features.getOrElse(AuthorIdFeature, None)).distinct

Stitch

.collect {

authorIds.map { authorId =>

getSuperFollowEligibilityOnUserClientColumn.fetcher

.fetch(authorId)

.map { authorId -> \_.v }

}

}.map { authorIdsToIsCreator =>

val authorIdsToIsCreatorMap = authorIdsToIsCreator.toMap

candidates.map { candidate =>

candidate.features.getOrElse(AuthorIdFeature, None) match {

case Some(authorId) =>

authorIdsToIsCreatorMap.get(authorId) match {

case Some(response) =>

keyFoundCounter.incr()

FeatureMapBuilder()

.add(AuthorIsCreatorFeature, response.getOrElse(false)).build()

case \_ =>

keyFailureCounter.incr()

DefaultFeatureMap

}

case \_ => MissingKeyFeatureMap

}

}

}

}

}

}