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

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.home\_mixer.product.scored\_tweets.feature\_hydrator.adapters.earlybird.EarlybirdAdapter

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

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

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

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

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

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

import com.twitter.home\_mixer.param.HomeMixerInjectionNames.EarlybirdRepository

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

import com.twitter.home\_mixer.util.earlybird.EarlybirdResponseUtil

import com.twitter.ml.api.DataRecord

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

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.FeatureWithDefaultOnFailure

import com.twitter.product\_mixer.core.feature.datarecord.DataRecordInAFeature

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.search.earlybird.{thriftscala => eb}

import com.twitter.servo.keyvalue.KeyValueResult

import com.twitter.servo.repository.KeyValueRepository

import com.twitter.stitch.Stitch

import com.twitter.util.Return

import javax.inject.Inject

import javax.inject.Named

import javax.inject.Singleton

import scala.collection.JavaConverters.\_

object EarlybirdDataRecordFeature

extends DataRecordInAFeature[TweetCandidate]

with FeatureWithDefaultOnFailure[TweetCandidate, DataRecord] {

override def defaultValue: DataRecord = new DataRecord()

}

@Singleton

class EarlybirdFeatureHydrator @Inject() (

@Named(EarlybirdRepository) client: KeyValueRepository[

(Seq[Long], Long),

Long,

eb.ThriftSearchResult

],

override val statsReceiver: StatsReceiver)

extends BulkCandidateFeatureHydrator[PipelineQuery, TweetCandidate]

with ObservedKeyValueResultHandler {

override val identifier: FeatureHydratorIdentifier = FeatureHydratorIdentifier("Earlybird")

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

EarlybirdDataRecordFeature,

EarlybirdFeature,

EarlybirdSearchResultFeature,

TweetUrlsFeature

)

override val statScope: String = identifier.toString

private val scopedStatsReceiver = statsReceiver.scope(statScope)

private val originalKeyFoundCounter = scopedStatsReceiver.counter("originalKey/found")

private val originalKeyLossCounter = scopedStatsReceiver.counter("originalKey/loss")

private val ebSearchResultNotExistPredicate: CandidateWithFeatures[TweetCandidate] => Boolean =

candidate => candidate.features.getOrElse(EarlybirdSearchResultFeature, None).isEmpty

private val ebFeaturesNotExistPredicate: CandidateWithFeatures[TweetCandidate] => Boolean =

candidate => candidate.features.getOrElse(EarlybirdFeature, None).isEmpty

override def apply(

query: PipelineQuery,

candidates: Seq[CandidateWithFeatures[TweetCandidate]]

): Stitch[Seq[FeatureMap]] = OffloadFuturePools.offloadFuture {

val candidatesToHydrate = candidates.filter { candidate =>

val isEmpty =

ebFeaturesNotExistPredicate(candidate) && ebSearchResultNotExistPredicate(candidate)

if (isEmpty) originalKeyLossCounter.incr() else originalKeyFoundCounter.incr()

isEmpty

}

client((candidatesToHydrate.map(\_.candidate.id), query.getRequiredUserId))

.map(handleResponse(query, candidates, \_, candidatesToHydrate))

}

private def handleResponse(

query: PipelineQuery,

candidates: Seq[CandidateWithFeatures[TweetCandidate]],

results: KeyValueResult[Long, eb.ThriftSearchResult],

candidatesToHydrate: Seq[CandidateWithFeatures[TweetCandidate]]

): Seq[FeatureMap] = {

val queryFeatureMap = query.features.getOrElse(FeatureMap.empty)

val userLanguages = queryFeatureMap.getOrElse(UserLanguagesFeature, Seq.empty)

val uiLanguageCode = queryFeatureMap.getOrElse(DeviceLanguageFeature, None)

val screenName = queryFeatureMap.getOrElse(UserScreenNameFeature, None)

val followedUserIds = queryFeatureMap.getOrElse(SGSFollowedUsersFeature, Seq.empty).toSet

val searchResults = candidatesToHydrate

.map { candidate =>

observedGet(Some(candidate.candidate.id), results)

}.collect {

case Return(Some(value)) => value

}

val allSearchResults = searchResults ++

candidates.filter(!ebSearchResultNotExistPredicate(\_)).flatMap { candidate =>

candidate.features

.getOrElse(EarlybirdSearchResultFeature, None)

}

val idToSearchResults = allSearchResults.map(sr => sr.id -> sr).toMap

val tweetIdToEbFeatures = EarlybirdResponseUtil.getTweetThriftFeaturesByTweetId(

searcherUserId = query.getRequiredUserId,

screenName = screenName,

userLanguages = userLanguages,

uiLanguageCode = uiLanguageCode,

followedUserIds = followedUserIds,

mutuallyFollowingUserIds = Set.empty,

searchResults = allSearchResults,

sourceTweetSearchResults = Seq.empty,

)

candidates.map { candidate =>

val transformedEbFeatures = tweetIdToEbFeatures.get(candidate.candidate.id)

val earlybirdFeatures =

if (transformedEbFeatures.nonEmpty) transformedEbFeatures

else candidate.features.getOrElse(EarlybirdFeature, None)

val candidateIsRetweet = candidate.features.getOrElse(IsRetweetFeature, false)

val sourceTweetEbFeatures =

candidate.features.getOrElse(SourceTweetEarlybirdFeature, None)

val originalTweetEbFeatures =

if (candidateIsRetweet && sourceTweetEbFeatures.nonEmpty)

sourceTweetEbFeatures

else earlybirdFeatures

val earlybirdDataRecord =

EarlybirdAdapter.adaptToDataRecords(originalTweetEbFeatures).asScala.head

FeatureMapBuilder()

.add(EarlybirdFeature, earlybirdFeatures)

.add(EarlybirdDataRecordFeature, earlybirdDataRecord)

.add(EarlybirdSearchResultFeature, idToSearchResults.get(candidate.candidate.id))

.add(TweetUrlsFeature, earlybirdFeatures.flatMap(\_.urlsList).getOrElse(Seq.empty))

.build()

}

}

}