package com.twitter.frigate.pushservice.adaptor

import com.twitter.explore\_ranker.thriftscala.ExploreRankerProductResponse

import com.twitter.explore\_ranker.thriftscala.ExploreRankerRequest

import com.twitter.explore\_ranker.thriftscala.ExploreRankerResponse

import com.twitter.explore\_ranker.thriftscala.ExploreRecommendation

import com.twitter.explore\_ranker.thriftscala.ImmersiveRecsResponse

import com.twitter.explore\_ranker.thriftscala.ImmersiveRecsResult

import com.twitter.explore\_ranker.thriftscala.NotificationsVideoRecs

import com.twitter.explore\_ranker.thriftscala.Product

import com.twitter.explore\_ranker.thriftscala.ProductContext

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.frigate.common.base.CandidateSource

import com.twitter.frigate.common.base.CandidateSourceEligible

import com.twitter.frigate.common.base.OutOfNetworkTweetCandidate

import com.twitter.frigate.pushservice.model.PushTypes.RawCandidate

import com.twitter.frigate.pushservice.model.PushTypes.Target

import com.twitter.frigate.pushservice.params.PushFeatureSwitchParams

import com.twitter.frigate.pushservice.util.AdaptorUtils

import com.twitter.frigate.pushservice.util.MediaCRT

import com.twitter.frigate.pushservice.util.PushAdaptorUtil

import com.twitter.frigate.pushservice.util.PushDeviceUtil

import com.twitter.frigate.thriftscala.CommonRecommendationType

import com.twitter.product\_mixer.core.thriftscala.ClientContext

import com.twitter.stitch.tweetypie.TweetyPie.TweetyPieResult

import com.twitter.storehaus.ReadableStore

import com.twitter.util.Future

case class ExploreVideoTweetCandidateAdaptor(

exploreRankerStore: ReadableStore[ExploreRankerRequest, ExploreRankerResponse],

tweetyPieStore: ReadableStore[Long, TweetyPieResult],

globalStats: StatsReceiver)

extends CandidateSource[Target, RawCandidate]

with CandidateSourceEligible[Target, RawCandidate] {

override def name: String = this.getClass.getSimpleName

private[this] val stats = globalStats.scope("ExploreVideoTweetCandidateAdaptor")

private[this] val totalInputRecs = stats.stat("input\_recs")

private[this] val totalRequests = stats.counter("total\_requests")

private[this] val totalEmptyResponse = stats.counter("total\_empty\_response")

private def buildExploreRankerRequest(

target: Target,

countryCode: Option[String],

language: Option[String],

): ExploreRankerRequest = {

ExploreRankerRequest(

clientContext = ClientContext(

userId = Some(target.targetId),

countryCode = countryCode,

languageCode = language,

),

product = Product.NotificationsVideoRecs,

productContext = Some(ProductContext.NotificationsVideoRecs(NotificationsVideoRecs())),

maxResults = Some(target.params(PushFeatureSwitchParams.MaxExploreVideoTweets))

)

}

override def get(target: Target): Future[Option[Seq[RawCandidate]]] = {

Future

.join(

target.countryCode,

target.inferredUserDeviceLanguage

).flatMap {

case (countryCode, language) =>

val request = buildExploreRankerRequest(target, countryCode, language)

exploreRankerStore.get(request).flatMap {

case Some(response) =>

val exploreResonseTweetIds = response match {

case ExploreRankerResponse(ExploreRankerProductResponse

.ImmersiveRecsResponse(ImmersiveRecsResponse(immersiveRecsResult))) =>

immersiveRecsResult.collect {

case ImmersiveRecsResult(ExploreRecommendation

.ExploreTweetRecommendation(exploreTweetRecommendation)) =>

exploreTweetRecommendation.tweetId

}

case \_ =>

Seq.empty

}

totalInputRecs.add(exploreResonseTweetIds.size)

totalRequests.incr()

AdaptorUtils

.getTweetyPieResults(exploreResonseTweetIds.toSet, tweetyPieStore).map {

tweetyPieResultMap =>

val candidates = tweetyPieResultMap.values.flatten

.map(buildVideoRawCandidates(target, \_))

Some(candidates.toSeq)

}

case \_ =>

totalEmptyResponse.incr()

Future.None

}

case \_ =>

Future.None

}

}

override def isCandidateSourceAvailable(target: Target): Future[Boolean] = {

PushDeviceUtil.isRecommendationsEligible(target).map { userRecommendationsEligible =>

userRecommendationsEligible && target.params(PushFeatureSwitchParams.EnableExploreVideoTweets)

}

}

private def buildVideoRawCandidates(

target: Target,

tweetyPieResult: TweetyPieResult

): RawCandidate with OutOfNetworkTweetCandidate = {

PushAdaptorUtil.generateOutOfNetworkTweetCandidates(

inputTarget = target,

id = tweetyPieResult.tweet.id,

mediaCRT = MediaCRT(

CommonRecommendationType.ExploreVideoTweet,

CommonRecommendationType.ExploreVideoTweet,

CommonRecommendationType.ExploreVideoTweet

),

result = Some(tweetyPieResult),

localizedEntity = None

)

}

}