package com.twitter.timelineranker.util

import com.twitter.search.common.features.thriftscala.ThriftTweetFeatures

import com.twitter.servo.util.FutureArrow

import com.twitter.timelineranker.core.HydratedCandidatesAndFeaturesEnvelope

import com.twitter.timelineranker.recap.model.ContentFeatures

import com.twitter.timelines.model.TweetId

import com.twitter.util.Future

/\*\*

\* Populates features with tweetId -> thriftTweetFeatures pairs.

\*

\* If a tweetId from contentFeatures is from searchResults, its content features are copied to

\* thriftTweetFeatures. If the tweet is a retweet, the original tweet's content features are copied.

\*

\* If the tweetId is not found in searchResults, but is an inReplyToTweet of a searchResult, the

\* tweetId -> thriftTweetFeatures pair is added to features. This is because in TLM, reply tweets

\* have features that are their inReplyToTweets' content features. This also allows scoring

\* inReplyToTweet with content features populated when scoring replies.

\*/

object CopyContentFeaturesIntoThriftTweetFeaturesTransform

extends FutureArrow[

HydratedCandidatesAndFeaturesEnvelope,

HydratedCandidatesAndFeaturesEnvelope

] {

override def apply(

request: HydratedCandidatesAndFeaturesEnvelope

): Future[HydratedCandidatesAndFeaturesEnvelope] = {

// Content Features Request Failures are handled in [[TweetypieContentFeaturesProvider]]

request.contentFeaturesFuture.map { contentFeaturesMap =>

val features = request.features.map {

case (tweetId: TweetId, thriftTweetFeatures: ThriftTweetFeatures) =>

val contentFeaturesOpt = request.tweetSourceTweetMap

.get(tweetId)

.orElse(

request.inReplyToTweetIds.contains(tweetId) match {

case true => Some(tweetId)

case false => None

}

)

.flatMap(contentFeaturesMap.get)

val thriftTweetFeaturesWithContentFeatures = contentFeaturesOpt match {

case Some(contentFeatures: ContentFeatures) =>

copyContentFeaturesIntoThriftTweetFeatures(contentFeatures, thriftTweetFeatures)

case \_ => thriftTweetFeatures

}

(tweetId, thriftTweetFeaturesWithContentFeatures)

}

request.copy(features = features)

}

}

def copyContentFeaturesIntoThriftTweetFeatures(

contentFeatures: ContentFeatures,

thriftTweetFeatures: ThriftTweetFeatures

): ThriftTweetFeatures = {

thriftTweetFeatures.copy(

tweetLength = Some(contentFeatures.length.toInt),

hasQuestion = Some(contentFeatures.hasQuestion),

numCaps = Some(contentFeatures.numCaps.toInt),

numWhitespaces = Some(contentFeatures.numWhiteSpaces.toInt),

numNewlines = contentFeatures.numNewlines,

videoDurationMs = contentFeatures.videoDurationMs,

bitRate = contentFeatures.bitRate,

aspectRatioNum = contentFeatures.aspectRatioNum,

aspectRatioDen = contentFeatures.aspectRatioDen,

widths = contentFeatures.widths.map(\_.map(\_.toInt)),

heights = contentFeatures.heights.map(\_.map(\_.toInt)),

resizeMethods = contentFeatures.resizeMethods.map(\_.map(\_.toInt)),

numMediaTags = contentFeatures.numMediaTags.map(\_.toInt),

mediaTagScreenNames = contentFeatures.mediaTagScreenNames,

emojiTokens = contentFeatures.emojiTokens,

emoticonTokens = contentFeatures.emoticonTokens,

phrases = contentFeatures.phrases,

textTokens = contentFeatures.tokens,

faceAreas = contentFeatures.faceAreas,

dominantColorRed = contentFeatures.dominantColorRed,

dominantColorBlue = contentFeatures.dominantColorBlue,

dominantColorGreen = contentFeatures.dominantColorGreen,

numColors = contentFeatures.numColors.map(\_.toInt),

stickerIds = contentFeatures.stickerIds,

mediaOriginProviders = contentFeatures.mediaOriginProviders,

isManaged = contentFeatures.isManaged,

is360 = contentFeatures.is360,

viewCount = contentFeatures.viewCount,

isMonetizable = contentFeatures.isMonetizable,

isEmbeddable = contentFeatures.isEmbeddable,

hasSelectedPreviewImage = contentFeatures.hasSelectedPreviewImage,

hasTitle = contentFeatures.hasTitle,

hasDescription = contentFeatures.hasDescription,

hasVisitSiteCallToAction = contentFeatures.hasVisitSiteCallToAction,

hasAppInstallCallToAction = contentFeatures.hasAppInstallCallToAction,

hasWatchNowCallToAction = contentFeatures.hasWatchNowCallToAction,

dominantColorPercentage = contentFeatures.dominantColorPercentage,

posUnigrams = contentFeatures.posUnigrams,

posBigrams = contentFeatures.posBigrams,

semanticCoreAnnotations = contentFeatures.semanticCoreAnnotations,

conversationControl = contentFeatures.conversationControl

)

}

}