package com.twitter.timelineranker.util

import com.twitter.servo.util.FutureArrow

import com.twitter.timelineranker.core.HydratedCandidatesAndFeaturesEnvelope

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

import com.twitter.timelines.model.tweet.HydratedTweet

import com.twitter.util.Future

object CopyContentFeaturesIntoHydratedTweetsTransform

extends FutureArrow[

HydratedCandidatesAndFeaturesEnvelope,

HydratedCandidatesAndFeaturesEnvelope

] {

override def apply(

request: HydratedCandidatesAndFeaturesEnvelope

): Future[HydratedCandidatesAndFeaturesEnvelope] = {

request.contentFeaturesFuture.map { sourceTweetContentFeaturesMap =>

val updatedHyratedTweets = request.candidateEnvelope.hydratedTweets.outerTweets.map {

hydratedTweet =>

val contentFeaturesOpt = request.tweetSourceTweetMap

.get(hydratedTweet.tweetId)

.flatMap(sourceTweetContentFeaturesMap.get)

val updatedHyratedTweet = contentFeaturesOpt match {

case Some(contentFeatures: ContentFeatures) =>

copyContentFeaturesIntoHydratedTweets(

contentFeatures,

hydratedTweet

)

case \_ => hydratedTweet

}

updatedHyratedTweet

}

request.copy(

candidateEnvelope = request.candidateEnvelope.copy(

hydratedTweets = request.candidateEnvelope.hydratedTweets.copy(

outerTweets = updatedHyratedTweets

)

)

)

}

}

def copyContentFeaturesIntoHydratedTweets(

contentFeatures: ContentFeatures,

hydratedTweet: HydratedTweet

): HydratedTweet = {

HydratedTweet(

hydratedTweet.tweet.copy(

selfThreadMetadata = contentFeatures.selfThreadMetadata,

media = contentFeatures.media

)

)

}

}