package com.twitter.timelineranker.util

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.servo.util.Gate

import com.twitter.timelineranker.contentfeatures.ContentFeaturesProvider

import com.twitter.timelineranker.core.HydratedTweets

import com.twitter.timelineranker.model.RecapQuery

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

import com.twitter.timelines.clients.tweetypie.TweetyPieClient

import com.twitter.timelines.model.TweetId

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

import com.twitter.timelines.util.FailOpenHandler

import com.twitter.tweetypie.thriftscala.MediaEntity

import com.twitter.tweetypie.thriftscala.TweetInclude

import com.twitter.tweetypie.thriftscala.{Tweet => TTweet}

import com.twitter.util.Future

object TweetypieContentFeaturesProvider {

val DefaultTweetyPieFieldsToHydrate: Set[TweetInclude] = TweetyPieClient.CoreTweetFields ++

TweetyPieClient.MediaFields ++

TweetyPieClient.SelfThreadFields ++

Set[TweetInclude](TweetInclude.MediaEntityFieldId(MediaEntity.AdditionalMetadataField.id))

//add Tweet fields from semantic core

val TweetyPieFieldsToHydrate: Set[TweetInclude] = DefaultTweetyPieFieldsToHydrate ++

Set[TweetInclude](TweetInclude.TweetFieldId(TTweet.EscherbirdEntityAnnotationsField.id))

val EmptyHydratedTweets: HydratedTweets =

HydratedTweets(Seq.empty[HydratedTweet], Seq.empty[HydratedTweet])

val EmptyHydratedTweetsFuture: Future[HydratedTweets] = Future.value(EmptyHydratedTweets)

}

class TweetypieContentFeaturesProvider(

tweetHydrator: TweetHydrator,

enableContentFeaturesGate: Gate[RecapQuery],

enableTokensInContentFeaturesGate: Gate[RecapQuery],

enableTweetTextInContentFeaturesGate: Gate[RecapQuery],

enableConversationControlContentFeaturesGate: Gate[RecapQuery],

enableTweetMediaHydrationGate: Gate[RecapQuery],

statsReceiver: StatsReceiver)

extends ContentFeaturesProvider {

val scopedStatsReceiver: StatsReceiver = statsReceiver.scope("TweetypieContentFeaturesProvider")

override def apply(

query: RecapQuery,

tweetIds: Seq[TweetId]

): Future[Map[TweetId, ContentFeatures]] = {

import TweetypieContentFeaturesProvider.\_

val tweetypieHydrationHandler = new FailOpenHandler(scopedStatsReceiver)

val hydratePenguinTextFeatures = enableContentFeaturesGate(query)

val hydrateSemanticCoreFeatures = enableContentFeaturesGate(query)

val hydrateTokens = enableTokensInContentFeaturesGate(query)

val hydrateTweetText = enableTweetTextInContentFeaturesGate(query)

val hydrateConversationControl = enableConversationControlContentFeaturesGate(query)

val userId = query.userId

val hydratedTweetsFuture = tweetypieHydrationHandler {

// tweetyPie fields to hydrate given hydrateSemanticCoreFeatures

val fieldsToHydrateWithSemanticCore = if (hydrateSemanticCoreFeatures) {

TweetyPieFieldsToHydrate

} else {

DefaultTweetyPieFieldsToHydrate

}

// tweetyPie fields to hydrate given hydrateSemanticCoreFeatures & hydrateConversationControl

val fieldsToHydrateWithConversationControl = if (hydrateConversationControl) {

fieldsToHydrateWithSemanticCore ++ TweetyPieClient.ConversationControlField

} else {

fieldsToHydrateWithSemanticCore

}

tweetHydrator.hydrate(Some(userId), tweetIds, fieldsToHydrateWithConversationControl)

} { e: Throwable => EmptyHydratedTweetsFuture }

hydratedTweetsFuture.map[Map[TweetId, ContentFeatures]] { hydratedTweets =>

hydratedTweets.outerTweets.map { hydratedTweet =>

val contentFeaturesFromTweet = ContentFeatures.Empty.copy(

selfThreadMetadata = hydratedTweet.tweet.selfThreadMetadata

)

val contentFeaturesWithText = TweetTextFeaturesExtractor.addTextFeaturesFromTweet(

contentFeaturesFromTweet,

hydratedTweet.tweet,

hydratePenguinTextFeatures,

hydrateTokens,

hydrateTweetText

)

val contentFeaturesWithMedia = TweetMediaFeaturesExtractor.addMediaFeaturesFromTweet(

contentFeaturesWithText,

hydratedTweet.tweet,

enableTweetMediaHydrationGate(query)

)

val contentFeaturesWithAnnotations = TweetAnnotationFeaturesExtractor

.addAnnotationFeaturesFromTweet(

contentFeaturesWithMedia,

hydratedTweet.tweet,

hydrateSemanticCoreFeatures

)

// add conversationControl to content features if hydrateConversationControl is true

if (hydrateConversationControl) {

val contentFeaturesWithConversationControl = contentFeaturesWithAnnotations.copy(

conversationControl = hydratedTweet.tweet.conversationControl

)

hydratedTweet.tweetId -> contentFeaturesWithConversationControl

} else {

hydratedTweet.tweetId -> contentFeaturesWithAnnotations

}

}.toMap

}

}

}