package com.twitter.home\_mixer.functional\_component.feature\_hydrator

import com.twitter.finagle.stats.StatsReceiver

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

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

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

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

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

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

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

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

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

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

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

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

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

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

import com.twitter.home\_mixer.model.request.FollowingProduct

import com.twitter.home\_mixer.model.request.ForYouProduct

import com.twitter.home\_mixer.model.request.ListTweetsProduct

import com.twitter.home\_mixer.model.request.ScoredTweetsProduct

import com.twitter.home\_mixer.model.request.SubscribedProduct

import com.twitter.home\_mixer.util.tweetypie.RequestFields

import com.twitter.product\_mixer.component\_library.feature\_hydrator.candidate.tweet\_is\_nsfw.IsNsfw

import com.twitter.product\_mixer.component\_library.feature\_hydrator.candidate.tweet\_visibility\_reason.VisibilityReason

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.featuremap.FeatureMap

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMapBuilder

import com.twitter.product\_mixer.core.functional\_component.feature\_hydrator.CandidateFeatureHydrator

import com.twitter.product\_mixer.core.model.common.identifier.FeatureHydratorIdentifier

import com.twitter.product\_mixer.core.pipeline.PipelineQuery

import com.twitter.spam.rtf.{thriftscala => rtf}

import com.twitter.stitch.Stitch

import com.twitter.stitch.tweetypie.{TweetyPie => TweetypieStitchClient}

import com.twitter.tweetypie.{thriftscala => tp}

import com.twitter.util.logging.Logging

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class TweetypieFeatureHydrator @Inject() (

tweetypieStitchClient: TweetypieStitchClient,

statsReceiver: StatsReceiver)

extends CandidateFeatureHydrator[PipelineQuery, TweetCandidate]

with Logging {

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

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

AuthorIdFeature,

ExclusiveConversationAuthorIdFeature,

InReplyToTweetIdFeature,

IsHydratedFeature,

IsNsfw,

IsNsfwFeature,

IsRetweetFeature,

QuotedTweetDroppedFeature,

QuotedTweetIdFeature,

QuotedUserIdFeature,

SourceTweetIdFeature,

SourceUserIdFeature,

TweetTextFeature,

TweetLanguageFeature,

VisibilityReason

)

private val DefaultFeatureMap = FeatureMapBuilder()

.add(IsHydratedFeature, false)

.add(IsNsfw, None)

.add(IsNsfwFeature, false)

.add(QuotedTweetDroppedFeature, false)

.add(TweetTextFeature, None)

.add(VisibilityReason, None)

.build()

override def apply(

query: PipelineQuery,

candidate: TweetCandidate,

existingFeatures: FeatureMap

): Stitch[FeatureMap] = {

val safetyLevel = query.product match {

case FollowingProduct => rtf.SafetyLevel.TimelineHomeLatest

case ForYouProduct =>

val inNetwork = existingFeatures.getOrElse(InNetworkFeature, true)

if (inNetwork) rtf.SafetyLevel.TimelineHome else rtf.SafetyLevel.TimelineHomeRecommendations

case ScoredTweetsProduct => rtf.SafetyLevel.TimelineHome

case ListTweetsProduct => rtf.SafetyLevel.TimelineLists

case SubscribedProduct => rtf.SafetyLevel.TimelineHomeSubscribed

case unknown => throw new UnsupportedOperationException(s"Unknown product: $unknown")

}

val tweetFieldsOptions = tp.GetTweetFieldsOptions(

tweetIncludes = RequestFields.TweetTPHydrationFields,

includeRetweetedTweet = true,

includeQuotedTweet = true,

visibilityPolicy = tp.TweetVisibilityPolicy.UserVisible,

safetyLevel = Some(safetyLevel),

forUserId = query.getOptionalUserId

)

val exclusiveAuthorIdOpt =

existingFeatures.getOrElse(ExclusiveConversationAuthorIdFeature, None)

tweetypieStitchClient.getTweetFields(tweetId = candidate.id, options = tweetFieldsOptions).map {

case tp.GetTweetFieldsResult(\_, tp.TweetFieldsResultState.Found(found), quoteOpt, \_) =>

val coreData = found.tweet.coreData

val isNsfwAdmin = coreData.exists(\_.nsfwAdmin)

val isNsfwUser = coreData.exists(\_.nsfwUser)

val quotedTweetDropped = quoteOpt.exists {

case \_: tp.TweetFieldsResultState.Filtered => true

case \_: tp.TweetFieldsResultState.NotFound => true

case \_ => false

}

val quotedTweetIsNsfw = quoteOpt.exists {

case quoteTweet: tp.TweetFieldsResultState.Found =>

quoteTweet.found.tweet.coreData.exists(data => data.nsfwAdmin || data.nsfwUser)

case \_ => false

}

val sourceTweetIsNsfw =

found.retweetedTweet.exists(\_.coreData.exists(data => data.nsfwAdmin || data.nsfwUser))

val tweetText = coreData.map(\_.text)

val tweetLanguage = found.tweet.language.map(\_.language)

val tweetAuthorId = coreData.map(\_.userId)

val inReplyToTweetId = coreData.flatMap(\_.reply.flatMap(\_.inReplyToStatusId))

val retweetedTweetId = found.retweetedTweet.map(\_.id)

val quotedTweetId = quoteOpt.flatMap {

case quoteTweet: tp.TweetFieldsResultState.Found =>

Some(quoteTweet.found.tweet.id)

case \_ => None

}

val retweetedTweetUserId = found.retweetedTweet.flatMap(\_.coreData).map(\_.userId)

val quotedTweetUserId = quoteOpt.flatMap {

case quoteTweet: tp.TweetFieldsResultState.Found =>

quoteTweet.found.tweet.coreData.map(\_.userId)

case \_ => None

}

val isNsfw = isNsfwAdmin || isNsfwUser || sourceTweetIsNsfw || quotedTweetIsNsfw

FeatureMapBuilder()

.add(AuthorIdFeature, tweetAuthorId)

.add(ExclusiveConversationAuthorIdFeature, exclusiveAuthorIdOpt)

.add(InReplyToTweetIdFeature, inReplyToTweetId)

.add(IsHydratedFeature, true)

.add(IsNsfw, Some(isNsfw))

.add(IsNsfwFeature, isNsfw)

.add(IsRetweetFeature, retweetedTweetId.isDefined)

.add(QuotedTweetDroppedFeature, quotedTweetDropped)

.add(QuotedTweetIdFeature, quotedTweetId)

.add(QuotedUserIdFeature, quotedTweetUserId)

.add(SourceTweetIdFeature, retweetedTweetId)

.add(SourceUserIdFeature, retweetedTweetUserId)

.add(TweetLanguageFeature, tweetLanguage)

.add(TweetTextFeature, tweetText)

.add(VisibilityReason, found.suppressReason)

.build()

// If no tweet result found, return default and pre-existing features

case \_ =>

DefaultFeatureMap ++ FeatureMapBuilder()

.add(AuthorIdFeature, existingFeatures.getOrElse(AuthorIdFeature, None))

.add(ExclusiveConversationAuthorIdFeature, exclusiveAuthorIdOpt)

.add(InReplyToTweetIdFeature, existingFeatures.getOrElse(InReplyToTweetIdFeature, None))

.add(IsRetweetFeature, existingFeatures.getOrElse(IsRetweetFeature, false))

.add(QuotedTweetIdFeature, existingFeatures.getOrElse(QuotedTweetIdFeature, None))

.add(QuotedUserIdFeature, existingFeatures.getOrElse(QuotedUserIdFeature, None))

.add(SourceTweetIdFeature, existingFeatures.getOrElse(SourceTweetIdFeature, None))

.add(SourceUserIdFeature, existingFeatures.getOrElse(SourceUserIdFeature, None))

.add(TweetLanguageFeature, existingFeatures.getOrElse(TweetLanguageFeature, None))

.build()

}

}

}