package com.twitter.product\_mixer.component\_library.feature\_hydrator.candidate.tweet\_tweetypie

import com.twitter.product\_mixer.component\_library.model.candidate.BaseTweetCandidate

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.SafetyLevel

import com.twitter.stitch.Stitch

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

import com.twitter.tweetypie.thriftscala.TweetVisibilityPolicy

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

// Candidate Features

object IsCommunityTweetFeature extends Feature[TweetCandidate, Boolean]

// Tweetypie VF Features

object HasTakedownFeature extends Feature[TweetCandidate, Boolean]

object HasTakedownForLocaleFeature extends Feature[TweetCandidate, Boolean]

object IsHydratedFeature extends Feature[TweetCandidate, Boolean]

object IsNarrowcastFeature extends Feature[TweetCandidate, Boolean]

object IsNsfwAdminFeature extends Feature[TweetCandidate, Boolean]

object IsNsfwFeature extends Feature[TweetCandidate, Boolean]

object IsNsfwUserFeature extends Feature[TweetCandidate, Boolean]

object IsNullcastFeature extends Feature[TweetCandidate, Boolean]

object QuotedTweetDroppedFeature extends Feature[TweetCandidate, Boolean]

object QuotedTweetHasTakedownFeature extends Feature[TweetCandidate, Boolean]

object QuotedTweetHasTakedownForLocaleFeature extends Feature[TweetCandidate, Boolean]

object QuotedTweetIdFeature extends Feature[TweetCandidate, Option[Long]]

object SourceTweetHasTakedownFeature extends Feature[TweetCandidate, Boolean]

object SourceTweetHasTakedownForLocaleFeature extends Feature[TweetCandidate, Boolean]

object TakedownCountryCodesFeature extends Feature[TweetCandidate, Set[String]]

object IsReplyFeature extends Feature[TweetCandidate, Boolean]

object InReplyToFeature extends Feature[TweetCandidate, Option[Long]]

object IsRetweetFeature extends Feature[TweetCandidate, Boolean]

object TweetTweetypieCandidateFeatureHydrator {

val CoreTweetFields: Set[TP.TweetInclude] = Set[TP.TweetInclude](

TP.TweetInclude.TweetFieldId(TP.Tweet.IdField.id),

TP.TweetInclude.TweetFieldId(TP.Tweet.CoreDataField.id)

)

val NsfwLabelFields: Set[TP.TweetInclude] = Set[TP.TweetInclude](

// Tweet fields containing NSFW related attributes, in addition to what exists in coreData.

TP.TweetInclude.TweetFieldId(TP.Tweet.NsfwHighRecallLabelField.id),

TP.TweetInclude.TweetFieldId(TP.Tweet.NsfwHighPrecisionLabelField.id),

TP.TweetInclude.TweetFieldId(TP.Tweet.NsfaHighRecallLabelField.id)

)

val SafetyLabelFields: Set[TP.TweetInclude] = Set[TP.TweetInclude](

// Tweet fields containing RTF labels for abuse and spam.

TP.TweetInclude.TweetFieldId(TP.Tweet.SpamLabelField.id),

TP.TweetInclude.TweetFieldId(TP.Tweet.AbusiveLabelField.id)

)

val OrganicTweetTPHydrationFields: Set[TP.TweetInclude] = CoreTweetFields ++

NsfwLabelFields ++

SafetyLabelFields ++

Set(

TP.TweetInclude.TweetFieldId(TP.Tweet.TakedownCountryCodesField.id),

// QTs imply a TweetyPie -> SGS request dependency

TP.TweetInclude.TweetFieldId(TP.Tweet.QuotedTweetField.id),

TP.TweetInclude.TweetFieldId(TP.Tweet.EscherbirdEntityAnnotationsField.id),

TP.TweetInclude.TweetFieldId(TP.Tweet.CommunitiesField.id),

// Field required for determining if a Tweet was created via News Camera.

TP.TweetInclude.TweetFieldId(TP.Tweet.ComposerSourceField.id)

)

val InjectedTweetTPHydrationFields: Set[TP.TweetInclude] =

OrganicTweetTPHydrationFields ++ Set(

// Mentions imply a TweetyPie -> Gizmoduck request dependency

TP.TweetInclude.TweetFieldId(TP.Tweet.MentionsField.id),

TP.TweetInclude.TweetFieldId(TP.Tweet.HashtagsField.id)

)

val DefaultFeatureMap = FeatureMapBuilder()

.add(IsNsfwAdminFeature, false)

.add(IsNsfwUserFeature, false)

.add(IsNsfwFeature, false)

.add(IsNullcastFeature, false)

.add(IsNarrowcastFeature, false)

.add(HasTakedownFeature, false)

.add(IsCommunityTweetFeature, false)

.add(TakedownCountryCodesFeature, Set.empty: Set[String])

.add(IsHydratedFeature, false)

.add(HasTakedownForLocaleFeature, false)

.add(QuotedTweetDroppedFeature, false)

.add(SourceTweetHasTakedownFeature, false)

.add(QuotedTweetHasTakedownFeature, false)

.add(SourceTweetHasTakedownForLocaleFeature, false)

.add(QuotedTweetHasTakedownForLocaleFeature, false)

.add(IsReplyFeature, false)

.add(InReplyToFeature, None)

.add(IsRetweetFeature, false)

.build()

}

class TweetTweetypieCandidateFeatureHydrator(

tweetypieStitchClient: TweetypieStitchClient,

safetyLevelPredicate: PipelineQuery => SafetyLevel)

extends CandidateFeatureHydrator[PipelineQuery, BaseTweetCandidate] {

import TweetTweetypieCandidateFeatureHydrator.\_

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

Set(

IsNsfwFeature,

IsNsfwAdminFeature,

IsNsfwUserFeature,

IsNullcastFeature,

IsNarrowcastFeature,

HasTakedownFeature,

IsCommunityTweetFeature,

TakedownCountryCodesFeature,

IsHydratedFeature,

HasTakedownForLocaleFeature,

QuotedTweetDroppedFeature,

SourceTweetHasTakedownFeature,

QuotedTweetHasTakedownFeature,

SourceTweetHasTakedownForLocaleFeature,

QuotedTweetHasTakedownForLocaleFeature,

IsReplyFeature,

InReplyToFeature,

IsRetweetFeature

)

override val identifier: FeatureHydratorIdentifier =

FeatureHydratorIdentifier("TweetTweetypie")

override def apply(

query: PipelineQuery,

candidate: BaseTweetCandidate,

existingFeatures: FeatureMap

): Stitch[FeatureMap] = {

val countryCode = query.getCountryCode.getOrElse("")

tweetypieStitchClient

.getTweetFields(

tweetId = candidate.id,

options = TP.GetTweetFieldsOptions(

tweetIncludes = OrganicTweetTPHydrationFields,

includeRetweetedTweet = true,

includeQuotedTweet = true,

visibilityPolicy = TweetVisibilityPolicy.UserVisible,

safetyLevel = Some(safetyLevelPredicate(query))

)

).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 hasTakedown = coreData.exists(\_.hasTakedown)

val isReply = coreData.exists(\_.reply.nonEmpty)

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

val isRetweet = coreData.exists(\_.share.nonEmpty)

val takedownCountryCodes =

found.tweet.takedownCountryCodes.getOrElse(Seq.empty).map(\_.toLowerCase).toSet

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 quotedTweetHasTakedown = quoteOpt.exists {

case quoteTweet: TP.TweetFieldsResultState.Found =>

quoteTweet.found.tweet.coreData.exists(\_.hasTakedown)

case \_ => false

}

val quotedTweetTakedownCountryCodes = quoteOpt

.collect {

case quoteTweet: TP.TweetFieldsResultState.Found =>

quoteTweet.found.tweet.takedownCountryCodes

.getOrElse(Seq.empty).map(\_.toLowerCase).toSet

}.getOrElse(Set.empty[String])

val sourceTweetIsNsfw =

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

val sourceTweetHasTakedown =

found.retweetedTweet.exists(\_.coreData.exists(\_.hasTakedown))

val sourceTweetTakedownCountryCodes = found.retweetedTweet

.map { sourceTweet: TP.Tweet =>

sourceTweet.takedownCountryCodes.getOrElse(Seq.empty).map(\_.toLowerCase).toSet

}.getOrElse(Set.empty)

FeatureMapBuilder()

.add(IsNsfwAdminFeature, isNsfwAdmin)

.add(IsNsfwUserFeature, isNsfwUser)

.add(IsNsfwFeature, isNsfwAdmin || isNsfwUser || sourceTweetIsNsfw || quotedTweetIsNsfw)

.add(IsNullcastFeature, coreData.exists(\_.nullcast))

.add(IsNarrowcastFeature, coreData.exists(\_.narrowcast.nonEmpty))

.add(HasTakedownFeature, hasTakedown)

.add(

HasTakedownForLocaleFeature,

hasTakedownForLocale(hasTakedown, countryCode, takedownCountryCodes))

.add(QuotedTweetDroppedFeature, quotedTweetDropped)

.add(SourceTweetHasTakedownFeature, sourceTweetHasTakedown)

.add(QuotedTweetHasTakedownFeature, quotedTweetHasTakedown)

.add(

SourceTweetHasTakedownForLocaleFeature,

hasTakedownForLocale(

sourceTweetHasTakedown,

countryCode,

sourceTweetTakedownCountryCodes))

.add(

QuotedTweetHasTakedownForLocaleFeature,

hasTakedownForLocale(

quotedTweetHasTakedown,

countryCode,

quotedTweetTakedownCountryCodes))

.add(IsCommunityTweetFeature, found.tweet.communities.exists(\_.communityIds.nonEmpty))

.add(

TakedownCountryCodesFeature,

found.tweet.takedownCountryCodes.getOrElse(Seq.empty).map(\_.toLowerCase).toSet)

.add(IsHydratedFeature, true)

.add(IsReplyFeature, isReply)

.add(InReplyToFeature, ancestorId)

.add(IsRetweetFeature, isRetweet)

.build()

// If no tweet result found, return default features

case \_ =>

DefaultFeatureMap

}

}

private def hasTakedownForLocale(

hasTakedown: Boolean,

countryCode: String,

takedownCountryCodes: Set[String]

) = hasTakedown && takedownCountryCodes.contains(countryCode)

}