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

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.FeatureWithDefaultOnFailure

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.BulkCandidateFeatureHydrator

import com.twitter.product\_mixer.core.model.common.CandidateWithFeatures

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

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

import com.twitter.stitch.Stitch

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

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

import com.twitter.util.Return

import com.twitter.util.Throw

import com.twitter.util.Try

import com.twitter.util.logging.Logging

// The VF NsfwHighPrecisionLabel that powers the NSFW determination here has been deprecated and is no longer written to.

@deprecated("Prefer VisibilityReason")

object IsNsfw extends FeatureWithDefaultOnFailure[TweetCandidate, Option[Boolean]] {

/\*\*

\* Generic Logic to evaluate whether a tweet is nsfw

\* @param hasNsfwHighPrecisionLabel flag for tweetypieTweet nsfwHighPrecision label

\* @param isNsfwUser flag for tweetypieTweet coreData nsfwUser flag

\* @param isNsfwAdmin flag for tweetypieTweet coreData nsfwAdmin flag

\* @return isNsfw to true if any of the three flags is true

\*/

def apply(

hasNsfwHighPrecisionLabel: Option[Boolean],

isNsfwUser: Option[Boolean],

isNsfwAdmin: Option[Boolean]

): Boolean = {

hasNsfwHighPrecisionLabel

.getOrElse(false) || (isNsfwUser.getOrElse(false) || isNsfwAdmin.getOrElse(false))

}

override val defaultValue = None

}

// The VF NsfwHighPrecisionLabel that powers the NSFW determination here has been deprecated and is no longer written to.

// TODO: Remove after all dependencies have migrated to using TweetCandidateVisibilityReasonFeatureHydrator.

@deprecated("Prefer TweetCandidateVisibilityReasonFeatureHydrator")

case class TweetIsNsfwCandidateFeatureHydrator(

tweetypieStitchClient: TweetypieStitchClient,

tweetVisibilityPolicy: t.TweetVisibilityPolicy)

extends BulkCandidateFeatureHydrator[PipelineQuery, BaseTweetCandidate]

with Logging {

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

override def features: Set[Feature[\_, \_]] = Set(IsNsfw)

private val NsfwLabelFields: Set[t.TweetInclude] = Set[t.TweetInclude](

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

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

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

)

override def apply(

query: PipelineQuery,

candidates: Seq[CandidateWithFeatures[BaseTweetCandidate]]

): Stitch[Seq[FeatureMap]] = {

Stitch

.traverse(candidates.map(\_.candidate.id)) { tweetId =>

tweetypieStitchClient

.getTweetFields(

tweetId = tweetId,

options = t.GetTweetFieldsOptions(

forUserId = query.getOptionalUserId,

tweetIncludes = NsfwLabelFields,

doNotCache = true,

visibilityPolicy = tweetVisibilityPolicy,

safetyLevel = None,

)

).liftToTry

}.map { getTweetFieldsResults: Seq[Try[t.GetTweetFieldsResult]] =>

val tweets: Seq[Try[t.Tweet]] = getTweetFieldsResults.map {

case Return(t.GetTweetFieldsResult(\_, t.TweetFieldsResultState.Found(found), \_, \_)) =>

Return(found.tweet)

case Return(t.GetTweetFieldsResult(\_, resultState, \_, \_)) =>

Throw(IsNsfwFeatureHydrationFailure(s"Unexpected tweet result state: ${resultState}"))

case Throw(e) =>

Throw(e)

}

candidates.zip(tweets).map {

case (candidateWithFeatures, tweetTry) =>

val isNsfwFeature = tweetTry.map { tweet =>

IsNsfw(

hasNsfwHighPrecisionLabel = Some(tweet.nsfwHighPrecisionLabel.isDefined),

isNsfwUser = tweet.coreData.map(\_.nsfwUser),

isNsfwAdmin = tweet.coreData.map(\_.nsfwAdmin)

)

}

FeatureMapBuilder()

.add(IsNsfw, isNsfwFeature.map(Some(\_)))

.build()

}

}

}

}

case class IsNsfwFeatureHydrationFailure(message: String)

extends Exception(s"IsNsfwFeatureHydrationFailure(${message})")