package com.twitter.visibility.builder.tweets

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.stitch.Stitch

import com.twitter.tweetypie.thriftscala.Tweet

import com.twitter.visibility.builder.FeatureMapBuilder

import com.twitter.visibility.common.TweetPerspectiveSource

import com.twitter.visibility.features.ViewerReportedTweet

class TweetPerspectiveFeatures(

tweetPerspectiveSource: TweetPerspectiveSource,

statsReceiver: StatsReceiver) {

private[this] val scopedStatsReceiver = statsReceiver.scope("tweet\_perspective\_features")

private[this] val reportedStats = scopedStatsReceiver.scope("reported")

def forTweet(

tweet: Tweet,

viewerId: Option[Long],

enableFetchReportedPerspective: Boolean

): FeatureMapBuilder => FeatureMapBuilder =

\_.withFeature(

ViewerReportedTweet,

tweetIsReported(tweet, viewerId, enableFetchReportedPerspective))

private[builder] def tweetIsReported(

tweet: Tweet,

viewerId: Option[Long],

enableFetchReportedPerspective: Boolean = true

): Stitch[Boolean] = {

((tweet.perspective, viewerId) match {

case (Some(perspective), \_) =>

Stitch.value(perspective.reported).onSuccess { \_ =>

reportedStats.counter("already\_hydrated").incr()

}

case (None, Some(viewerId)) =>

if (enableFetchReportedPerspective) {

tweetPerspectiveSource.reported(tweet.id, viewerId).onSuccess { \_ =>

reportedStats.counter("request").incr()

}

} else {

Stitch.False.onSuccess { \_ =>

reportedStats.counter("light\_request").incr()

}

}

case \_ =>

Stitch.False.onSuccess { \_ =>

reportedStats.counter("empty").incr()

}

}).onSuccess { \_ =>

reportedStats.counter("").incr()

}

}

}