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

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

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

import com.twitter.product\_mixer.component\_library.feature\_hydrator.query.social\_graph.SGSFollowedUsersFeature

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.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

object InNetworkFeatureHydrator

extends BulkCandidateFeatureHydrator[PipelineQuery, TweetCandidate] {

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

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

override def apply(

query: PipelineQuery,

candidates: Seq[CandidateWithFeatures[TweetCandidate]]

): Stitch[Seq[FeatureMap]] = {

val viewerId = query.getRequiredUserId

val followedUserIds = query.features.get.get(SGSFollowedUsersFeature).toSet

val featureMaps = candidates.map { candidate =>

// We use authorId and not sourceAuthorId here so that retweets are defined as in network

val isInNetworkOpt = candidate.features.getOrElse(AuthorIdFeature, None).map { authorId =>

// Users cannot follow themselves but this is in network by definition

val isSelfTweet = authorId == viewerId

isSelfTweet || followedUserIds.contains(authorId)

}

FeatureMapBuilder().add(InNetworkFeature, isInNetworkOpt.getOrElse(true)).build()

}

Stitch.value(featureMaps)

}

}