package com.twitter.home\_mixer.product.scored\_tweets.gate

import com.twitter.home\_mixer.product.scored\_tweets.gate.MinCachedTweetsGate.identifierSuffix

import com.twitter.home\_mixer.util.CachedScoredTweetsHelper

import com.twitter.product\_mixer.core.functional\_component.gate.Gate

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

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

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

import com.twitter.stitch.Stitch

import com.twitter.timelines.configapi.Param

case class MinCachedTweetsGate(

candidatePipelineIdentifier: CandidatePipelineIdentifier,

minCachedTweetsParam: Param[Int])

extends Gate[PipelineQuery] {

override val identifier: GateIdentifier =

GateIdentifier(candidatePipelineIdentifier + identifierSuffix)

override def shouldContinue(query: PipelineQuery): Stitch[Boolean] = {

val minCachedTweets = query.params(minCachedTweetsParam)

val cachedScoredTweets =

query.features.map(CachedScoredTweetsHelper.unseenCachedScoredTweets).getOrElse(Seq.empty)

val numCachedTweets = cachedScoredTweets.count { tweet =>

tweet.candidatePipelineIdentifier.exists(

CandidatePipelineIdentifier(\_).equals(candidatePipelineIdentifier))

}

Stitch.value(numCachedTweets < minCachedTweets)

}

}

object MinCachedTweetsGate {

val identifierSuffix = "MinCachedTweets"

}