package com.twitter.timelineranker.common

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.servo.util.FutureArrow

import com.twitter.timelineranker.core.CandidateEnvelope

import com.twitter.timelines.clients.relevance\_search.SearchClient

import com.twitter.util.Future

object OutOfNetworkRepliesToUserIdSearchResultsTransform {

val DefaultMaxTweetCount = 100

}

// Requests search results for out-of-network replies to a user Id

class OutOfNetworkRepliesToUserIdSearchResultsTransform(

searchClient: SearchClient,

statsReceiver: StatsReceiver,

logSearchDebugInfo: Boolean = true)

extends FutureArrow[CandidateEnvelope, CandidateEnvelope] {

private[this] val maxCountStat = statsReceiver.stat("maxCount")

private[this] val numResultsFromSearchStat = statsReceiver.stat("numResultsFromSearch")

private[this] val earlybirdScoreX100Stat = statsReceiver.stat("earlybirdScoreX100")

override def apply(envelope: CandidateEnvelope): Future[CandidateEnvelope] = {

val maxCount = envelope.query.maxCount

.getOrElse(OutOfNetworkRepliesToUserIdSearchResultsTransform.DefaultMaxTweetCount)

maxCountStat.add(maxCount)

envelope.followGraphData.followedUserIdsFuture

.flatMap {

case followedIds =>

searchClient

.getOutOfNetworkRepliesToUserId(

userId = envelope.query.userId,

followedUserIds = followedIds.toSet,

maxCount = maxCount,

earlybirdOptions = envelope.query.earlybirdOptions,

logSearchDebugInfo

).map { results =>

numResultsFromSearchStat.add(results.size)

results.foreach { result =>

val earlybirdScoreX100 =

result.metadata.flatMap(\_.score).getOrElse(0.0).toFloat \* 100

earlybirdScoreX100Stat.add(earlybirdScoreX100)

}

envelope.copy(searchResults = results)

}

}

}

}