package com.twitter.timelineranker.monitoring

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.search.earlybird.thriftscala.ThriftSearchResult

import com.twitter.servo.util.FutureArrow

import com.twitter.timelineranker.core.CandidateEnvelope

import com.twitter.timelineranker.model.RecapQuery.DependencyProvider

import com.twitter.util.Future

/\*\*

\* Captures tweet counts pre and post transformation for a list of users

\*/

class UsersSearchResultMonitoringTransform(

name: String,

underlyingTransformer: FutureArrow[CandidateEnvelope, CandidateEnvelope],

statsReceiver: StatsReceiver,

debugAuthorListDependencyProvider: DependencyProvider[Seq[Long]])

extends FutureArrow[CandidateEnvelope, CandidateEnvelope] {

private val scopedStatsReceiver = statsReceiver.scope(name)

private val preTransformCounter = (userId: Long) =>

scopedStatsReceiver

.scope("pre\_transform").scope(userId.toString).counter("debug\_author\_list")

private val postTransformCounter = (userId: Long) =>

scopedStatsReceiver

.scope("post\_transform").scope(userId.toString).counter("debug\_author\_list")

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

val debugAuthorList = debugAuthorListDependencyProvider.apply(envelope.query)

envelope.searchResults

.filter(isTweetFromDebugAuthorList(\_, debugAuthorList))

.flatMap(\_.metadata)

.foreach(metadata => preTransformCounter(metadata.fromUserId).incr())

underlyingTransformer

.apply(envelope)

.map { result =>

envelope.searchResults

.filter(isTweetFromDebugAuthorList(\_, debugAuthorList))

.flatMap(\_.metadata)

.foreach(metadata => postTransformCounter(metadata.fromUserId).incr())

result

}

}

private def isTweetFromDebugAuthorList(

searchResult: ThriftSearchResult,

debugAuthorList: Seq[Long]

): Boolean =

searchResult.metadata.exists(metadata => debugAuthorList.contains(metadata.fromUserId))

}