package com.twitter.frigate.pushservice.rank

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.frigate.common.base.CandidateDetails

import com.twitter.frigate.common.base.TweetCandidate

import com.twitter.frigate.pushservice.model.PushTypes.PushCandidate

import com.twitter.stitch.tweetypie.TweetyPie.TweetyPieResult

import com.twitter.storehaus.ReadableStore

import com.twitter.util.Future

class LoggedOutRanker(tweetyPieStore: ReadableStore[Long, TweetyPieResult], stats: StatsReceiver) {

private val statsReceiver = stats.scope(this.getClass.getSimpleName)

private val rankedCandidates = statsReceiver.counter("ranked\_candidates\_count")

def rank(

candidates: Seq[CandidateDetails[PushCandidate]]

): Future[Seq[CandidateDetails[PushCandidate]]] = {

val tweetIds = candidates.map { cand => cand.candidate.asInstanceOf[TweetCandidate].tweetId }

val results = tweetyPieStore.multiGet(tweetIds.toSet).values.toSeq

val futureOfResults = Future.traverseSequentially(results)(r => r)

val tweetsFut = futureOfResults.map { tweetyPieResults =>

tweetyPieResults.map(\_.map(\_.tweet))

}

val sortedTweetsFuture = tweetsFut.map { tweets =>

tweets

.map { tweet =>

if (tweet.isDefined && tweet.get.counts.isDefined) {

tweet.get.id -> tweet.get.counts.get.favoriteCount.getOrElse(0L)

} else {

0 -> 0L

}

}.sortBy(\_.\_2)(Ordering[Long].reverse)

}

val finalCandidates = sortedTweetsFuture.map { sortedTweets =>

sortedTweets

.map { tweet =>

candidates.find(\_.candidate.asInstanceOf[TweetCandidate].tweetId == tweet.\_1).orNull

}.filter { cand => cand != null }

}

finalCandidates.map { fc =>

rankedCandidates.incr(fc.size)

}

finalCandidates

}

}