package com.twitter.timelineranker.entity\_tweets

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.servo.util.FutureArrow

import com.twitter.timelineranker.core.CandidateEnvelope

import com.twitter.timelineranker.model.TweetIdRange

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

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

import com.twitter.timelines.model.TweetId

import com.twitter.util.Future

object EntityTweetsSearchResultsTransform {

// If EntityTweetsQuery.maxCount is not specified, the following count is used.

val DefaultEntityTweetsMaxTweetCount = 200

}

/\*\*

\* Fetch entity tweets search results using the search client

\* and populate them into the CandidateEnvelope

\*/

class EntityTweetsSearchResultsTransform(

searchClient: SearchClient,

statsReceiver: StatsReceiver,

logSearchDebugInfo: Boolean = false)

extends FutureArrow[CandidateEnvelope, CandidateEnvelope] {

import EntityTweetsSearchResultsTransform.\_

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

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

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

val maxCount = envelope.query.maxCount.getOrElse(DefaultEntityTweetsMaxTweetCount)

maxCountStat.add(maxCount)

val tweetIdRange = envelope.query.range

.map(TweetIdRange.fromTimelineRange)

.getOrElse(TweetIdRange.default)

val beforeTweetIdExclusive = tweetIdRange.toId

val afterTweetIdExclusive = tweetIdRange.fromId

val excludedTweetIds = envelope.query.excludedTweetIds.getOrElse(Seq.empty[TweetId]).toSet

val languages = envelope.query.languages.map(\_.map(\_.language))

envelope.followGraphData.inNetworkUserIdsFuture.flatMap { inNetworkUserIds =>

searchClient

.getEntityTweets(

userId = Some(envelope.query.userId),

followedUserIds = inNetworkUserIds.toSet,

maxCount = maxCount,

beforeTweetIdExclusive = beforeTweetIdExclusive,

afterTweetIdExclusive = afterTweetIdExclusive,

earlybirdOptions = envelope.query.earlybirdOptions,

semanticCoreIds = envelope.query.semanticCoreIds,

hashtags = envelope.query.hashtags,

languages = languages,

tweetTypes = TweetTypes.fromTweetKindOption(envelope.query.options),

searchOperator = envelope.query.searchOperator,

excludedTweetIds = excludedTweetIds,

logSearchDebugInfo = logSearchDebugInfo,

includeNullcastTweets = envelope.query.includeNullcastTweets.getOrElse(false),

includeTweetsFromArchiveIndex =

envelope.query.includeTweetsFromArchiveIndex.getOrElse(false),

authorIds = envelope.query.authorIds.map(\_.toSet)

).map { results =>

numResultsFromSearchStat.add(results.size)

envelope.copy(searchResults = results)

}

}

}

}