package com.twitter.timelineranker.util

import com.twitter.finagle.stats.StatsReceiver

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

import com.twitter.timelines.model.TweetId

import com.twitter.timelines.model.UserId

object SourceTweetsUtil {

def getSourceTweetIds(

searchResults: Seq[ThriftSearchResult],

searchResultsTweetIds: Set[TweetId],

followedUserIds: Seq[TweetId],

shouldIncludeReplyRootTweets: Boolean,

statsReceiver: StatsReceiver

): Seq[TweetId] = {

val replyRootTweetCounter = statsReceiver.counter("replyRootTweet")

val retweetSourceTweetIds = getRetweetSourceTweetIds(searchResults, searchResultsTweetIds)

val inNetworkReplyInReplyToTweetIds = getInNetworkInReplyToTweetIds(

searchResults,

searchResultsTweetIds,

followedUserIds

)

val extendedRepliesSourceTweetIds = getExtendedReplySourceTweetIds(

searchResults,

searchResultsTweetIds,

followedUserIds

)

val replyRootTweetIds = if (shouldIncludeReplyRootTweets) {

val rootTweetIds = getReplyRootTweetIds(

searchResults,

searchResultsTweetIds

)

replyRootTweetCounter.incr(rootTweetIds.size)

rootTweetIds

} else {

Seq.empty

}

(retweetSourceTweetIds ++ extendedRepliesSourceTweetIds ++

inNetworkReplyInReplyToTweetIds ++ replyRootTweetIds).distinct

}

def getInNetworkInReplyToTweetIds(

searchResults: Seq[ThriftSearchResult],

searchResultsTweetIds: Set[TweetId],

followedUserIds: Seq[UserId]

): Seq[TweetId] = {

searchResults

.filter(SearchResultUtil.isInNetworkReply(followedUserIds))

.flatMap(SearchResultUtil.getSourceTweetId)

.filterNot(searchResultsTweetIds.contains)

}

def getReplyRootTweetIds(

searchResults: Seq[ThriftSearchResult],

searchResultsTweetIds: Set[TweetId]

): Seq[TweetId] = {

searchResults

.flatMap(SearchResultUtil.getReplyRootTweetId)

.filterNot(searchResultsTweetIds.contains)

}

def getRetweetSourceTweetIds(

searchResults: Seq[ThriftSearchResult],

searchResultsTweetIds: Set[TweetId]

): Seq[TweetId] = {

searchResults

.filter(SearchResultUtil.isRetweet)

.flatMap(SearchResultUtil.getSourceTweetId)

.filterNot(searchResultsTweetIds.contains)

}

def getExtendedReplySourceTweetIds(

searchResults: Seq[ThriftSearchResult],

searchResultsTweetIds: Set[TweetId],

followedUserIds: Seq[UserId]

): Seq[TweetId] = {

searchResults

.filter(SearchResultUtil.isExtendedReply(followedUserIds))

.flatMap(SearchResultUtil.getSourceTweetId)

.filterNot(searchResultsTweetIds.contains)

}

}