package com.twitter.timelineranker.common

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.timelineranker.util.SourceTweetsUtil

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

import com.twitter.timelines.util.FailOpenHandler

import com.twitter.util.Future

object SourceTweetsSearchResultsTransform {

val EmptySearchResults: Seq[ThriftSearchResult] = Seq.empty[ThriftSearchResult]

val EmptySearchResultsFuture: Future[Seq[ThriftSearchResult]] = Future.value(EmptySearchResults)

}

/\*\*

\* Fetch source tweets for a given set of search results

\* Collects ids of source tweets, including extended reply and reply source tweets if needed,

\* fetches those tweets from search and populates them into the envelope

\*/

class SourceTweetsSearchResultsTransform(

searchClient: SearchClient,

failOpenHandler: FailOpenHandler,

hydrateReplyRootTweetProvider: DependencyProvider[Boolean],

perRequestSourceSearchClientIdProvider: DependencyProvider[Option[String]],

statsReceiver: StatsReceiver)

extends FutureArrow[CandidateEnvelope, CandidateEnvelope] {

import SourceTweetsSearchResultsTransform.\_

private val scopedStatsReceiver = statsReceiver.scope(getClass.getSimpleName)

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

failOpenHandler {

envelope.followGraphData.followedUserIdsFuture.flatMap { followedUserIds =>

// NOTE: tweetIds are pre-computed as a performance optimisation.

val searchResultsTweetIds = envelope.searchResults.map(\_.id).toSet

val sourceTweetIds = SourceTweetsUtil.getSourceTweetIds(

searchResults = envelope.searchResults,

searchResultsTweetIds = searchResultsTweetIds,

followedUserIds = followedUserIds,

shouldIncludeReplyRootTweets = hydrateReplyRootTweetProvider(envelope.query),

statsReceiver = scopedStatsReceiver

)

if (sourceTweetIds.isEmpty) {

EmptySearchResultsFuture

} else {

searchClient.getTweetsScoredForRecap(

userId = envelope.query.userId,

tweetIds = sourceTweetIds,

earlybirdOptions = envelope.query.earlybirdOptions,

logSearchDebugInfo = false,

searchClientId = perRequestSourceSearchClientIdProvider(envelope.query)

)

}

}

} { \_: Throwable => EmptySearchResultsFuture }.map { sourceSearchResults =>

envelope.copy(sourceSearchResults = sourceSearchResults)

}

}

}