package com.twitter.timelineranker.common

import com.twitter.finagle.stats.StatsReceiver

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.util.Future

/\*\*

\* trims elements of the envelope other than the searchResults

\* (i.e. sourceSearchResults, hydratedTweets, sourceHydratedTweets) to match with searchResults.

\*/

class TrimToMatchSearchResultsTransform(

hydrateReplyRootTweetProvider: DependencyProvider[Boolean],

statsReceiver: StatsReceiver)

extends FutureArrow[CandidateEnvelope, CandidateEnvelope] {

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

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

val searchResults = envelope.searchResults

val searchResultsIds = searchResults.map(\_.id).toSet

// Trim rest of the seqs to match top search results.

val hydratedTweets = envelope.hydratedTweets.outerTweets

val topHydratedTweets = hydratedTweets.filter(ht => searchResultsIds.contains(ht.tweetId))

envelope.followGraphData.followedUserIdsFuture.map { followedUserIds =>

val sourceTweetIdsOfTopResults =

SourceTweetsUtil

.getSourceTweetIds(

searchResults = searchResults,

searchResultsTweetIds = searchResultsIds,

followedUserIds = followedUserIds,

shouldIncludeReplyRootTweets = hydrateReplyRootTweetProvider(envelope.query),

statsReceiver = scopedStatsReceiver

).toSet

val sourceTweetSearchResultsForTopN =

envelope.sourceSearchResults.filter(r => sourceTweetIdsOfTopResults.contains(r.id))

val hydratedSourceTweetsForTopN =

envelope.sourceHydratedTweets.outerTweets.filter(ht =>

sourceTweetIdsOfTopResults.contains(ht.tweetId))

val hydratedTweetsForEnvelope = envelope.hydratedTweets.copy(outerTweets = topHydratedTweets)

val hydratedSourceTweetsForEnvelope =

envelope.sourceHydratedTweets.copy(outerTweets = hydratedSourceTweetsForTopN)

envelope.copy(

hydratedTweets = hydratedTweetsForEnvelope,

searchResults = searchResults,

sourceHydratedTweets = hydratedSourceTweetsForEnvelope,

sourceSearchResults = sourceTweetSearchResultsForTopN

)

}

}

}