package com.twitter.timelineranker.common

import com.twitter.servo.util.FutureArrow

import com.twitter.timelineranker.core.CandidateEnvelope

import com.twitter.timelineranker.model.CandidateTweet

import com.twitter.timelineranker.model.RecapQuery.DependencyProvider

import com.twitter.util.Future

import com.twitter.util.Time

import scala.util.Random

/\*\*

\* picks up one or more random tweets and sets its tweetFeatures.isRandomTweet field to true.

\*/

class MarkRandomTweetTransform(

includeRandomTweetProvider: DependencyProvider[Boolean],

randomGenerator: Random = new Random(Time.now.inMilliseconds),

includeSingleRandomTweetProvider: DependencyProvider[Boolean],

probabilityRandomTweetProvider: DependencyProvider[Double])

extends FutureArrow[CandidateEnvelope, CandidateEnvelope] {

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

val includeRandomTweet = includeRandomTweetProvider(envelope.query)

val includeSingleRandomTweet = includeSingleRandomTweetProvider(envelope.query)

val probabilityRandomTweet = probabilityRandomTweetProvider(envelope.query)

val searchResults = envelope.searchResults

if (!includeRandomTweet || searchResults.isEmpty) { // random tweet off

Future.value(envelope)

} else if (includeSingleRandomTweet) { // pick only one

val randomIdx = randomGenerator.nextInt(searchResults.size)

val randomTweet = searchResults(randomIdx)

val randomTweetWithFlag = randomTweet.copy(

tweetFeatures = randomTweet.tweetFeatures

.orElse(Some(CandidateTweet.DefaultFeatures))

.map(\_.copy(isRandomTweet = Some(true)))

)

val updatedSearchResults = searchResults.updated(randomIdx, randomTweetWithFlag)

Future.value(envelope.copy(searchResults = updatedSearchResults))

} else { // pick tweets with perTweetProbability

val updatedSearchResults = searchResults.map { result =>

if (randomGenerator.nextDouble() < probabilityRandomTweet) {

result.copy(

tweetFeatures = result.tweetFeatures

.orElse(Some(CandidateTweet.DefaultFeatures))

.map(\_.copy(isRandomTweet = Some(true))))

} else

result

}

Future.value(envelope.copy(searchResults = updatedSearchResults))

}

}

}