package com.twitter.timelineranker.uteg\_liked\_by\_tweets

import com.twitter.recos.recos\_common.thriftscala.SocialProofType

import com.twitter.recos.user\_tweet\_entity\_graph.thriftscala.TweetRecommendation

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.timelines.model.TweetId

import com.twitter.util.Future

class MinNumNonAuthorFavoritedByUserIdsFilterTransform(

minNumFavoritedByUserIdsProvider: DependencyProvider[Int])

extends FutureArrow[CandidateEnvelope, CandidateEnvelope] {

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

val filteredSearchResults = envelope.searchResults.filter { searchResult =>

numNonAuthorFavs(

searchResult = searchResult,

utegResultsMap = envelope.utegResults

).exists(\_ >= minNumFavoritedByUserIdsProvider(envelope.query))

}

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

}

// return number of non-author users that faved the tweet in a searchResult

// return None if author is None or if the tweet is not found in utegResultsMap

protected def numNonAuthorFavs(

searchResult: ThriftSearchResult,

utegResultsMap: Map[TweetId, TweetRecommendation]

): Option[Int] = {

for {

metadata <- searchResult.metadata

authorId = metadata.fromUserId

tweetRecommendation <- utegResultsMap.get(searchResult.id)

favedByUserIds <- tweetRecommendation.socialProofByType.get(SocialProofType.Favorite)

} yield favedByUserIds.filterNot(\_ == authorId).size

}

}