package com.twitter.recos.user\_tweet\_graph.util

import com.twitter.graphjet.bipartite.api.BipartiteGraph

import com.twitter.recos.user\_tweet\_graph.thriftscala.\_

import com.twitter.recos.features.tweet.thriftscala.GraphFeaturesForTweet

import com.twitter.graphjet.algorithms.TweetIDMask

object GetRelatedTweetCandidatesUtil {

private val tweetIDMask = new TweetIDMask

/\*\*

\* calculate scores for each RHS tweet that we get back

\* for tweetBasedRelatedTweet, scorePreFactor = queryTweetDegree / log(queryTweetDegree) / LHSuserSize

\* and the final score will be a log-cosine score

\* for non-tweetBasedRelatedTweet, We don't have a query tweet, to keep scoring function consistent,

\* scorePreFactor = 1000.0 / LHSuserSize (queryTweetDegree's average is ~10k, 1000 ~= 10k/log(10k))

\* Though scorePreFactor is applied for all results within a request, it's still useful to make score comparable across requests,

\* so we can have a unifed min\_score and help with downstream score normalization

\* \*\*/

def getRelatedTweetCandidates(

relatedTweetCandidates: Seq[Long],

minCooccurrence: Int,

minResultDegree: Int,

scorePreFactor: Double,

bipartiteGraph: BipartiteGraph,

): Seq[RelatedTweet] = {

relatedTweetCandidates

.groupBy(tweetId => tweetId)

.filterKeys(tweetId => bipartiteGraph.getRightNodeDegree(tweetId) > minResultDegree)

.mapValues(\_.size)

.filter { case (\_, cooccurrence) => cooccurrence >= minCooccurrence }

.toSeq

.map {

case (relatedTweetId, cooccurrence) =>

val relatedTweetDegree = bipartiteGraph.getRightNodeDegree(relatedTweetId)

val score = scorePreFactor \* cooccurrence / math.log(relatedTweetDegree)

toRelatedTweet(relatedTweetId, score, relatedTweetDegree, cooccurrence)

}

.sortBy(-\_.score)

}

def toRelatedTweet(

relatedTweetId: Long,

score: Double,

relatedTweetDegree: Int,

cooccurrence: Int

): RelatedTweet = {

RelatedTweet(

tweetId = tweetIDMask.restore(relatedTweetId),

score = score,

relatedTweetGraphFeatures = Some(

GraphFeaturesForTweet(cooccurrence = Some(cooccurrence), degree = Some(relatedTweetDegree)))

)

}

}