package com.twitter.recosinjector.edges

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.graphjet.algorithms.RecommendationType

import com.twitter.recosinjector.clients.CacheEntityEntry

import com.twitter.recosinjector.clients.RecosHoseEntitiesCache

import com.twitter.recosinjector.clients.UrlResolver

import com.twitter.recosinjector.util.TweetDetails

import com.twitter.util.Future

import scala.collection.Map

import scala.util.hashing.MurmurHash3

class UserTweetEntityEdgeBuilder(

cache: RecosHoseEntitiesCache,

urlResolver: UrlResolver

)(

implicit val stats: StatsReceiver) {

def getHashedEntities(entities: Seq[String]): Seq[Int] = {

entities.map(MurmurHash3.stringHash)

}

/\*\*

\* Given the entities and their corresponding hashedIds, store the hashId->entity mapping into a

\* cache.

\* This is because UTEG edges only store the hashIds, and relies on the cache values to

\* recover the actual entities. This allows us to store integer values instead of string in the

\* edges to save space.

\*/

private def storeEntitiesInCache(

urlEntities: Seq[String],

urlHashIds: Seq[Int]

): Future[Unit] = {

val urlCacheEntries = urlHashIds.zip(urlEntities).map {

case (hashId, url) =>

CacheEntityEntry(RecosHoseEntitiesCache.UrlPrefix, hashId, url)

}

cache.updateEntitiesCache(

newCacheEntries = urlCacheEntries,

stats = stats.scope("urlCache")

)

}

/\*\*

\* Return an entity mapping from GraphJet recType -> hash(entity)

\*/

private def getEntitiesMap(

urlHashIds: Seq[Int]

) = {

val entitiesMap = Seq(

RecommendationType.URL.getValue.toByte -> urlHashIds

).collect {

case (keys, ids) if ids.nonEmpty => keys -> ids

}.toMap

if (entitiesMap.isEmpty) None else Some(entitiesMap)

}

def getEntitiesMapAndUpdateCache(

tweetId: Long,

tweetDetails: Option[TweetDetails]

): Future[Option[Map[Byte, Seq[Int]]]] = {

val resolvedUrlFut = urlResolver

.getResolvedUrls(

urls = tweetDetails.flatMap(\_.urls).getOrElse(Nil),

tweetId = tweetId

).map(\_.values.toSeq)

resolvedUrlFut.map { resolvedUrls =>

val urlEntities = resolvedUrls

val urlHashIds = getHashedEntities(urlEntities)

// Async call to cache

storeEntitiesInCache(

urlEntities = urlEntities,

urlHashIds = urlHashIds

)

getEntitiesMap(urlHashIds)

}

}

}