package com.twitter.interaction\_graph.scio.agg\_address\_book

import com.spotify.scio.values.SCollection

import com.twitter.addressbook.matches.thriftscala.UserMatchesRecord

import com.twitter.interaction\_graph.scio.common.FeatureGeneratorUtil

import com.twitter.interaction\_graph.scio.common.InteractionGraphRawInput

import com.twitter.interaction\_graph.thriftscala.Edge

import com.twitter.interaction\_graph.thriftscala.FeatureName

import com.twitter.interaction\_graph.thriftscala.Vertex

object InteractionGraphAddressBookUtil {

val EMAIL = "email"

val PHONE = "phone"

val BOTH = "both"

val DefaultAge = 1

val DegaultFeatureValue = 1.0

def process(

addressBook: SCollection[UserMatchesRecord]

)(

implicit addressBookCounters: InteractionGraphAddressBookCountersTrait

): (SCollection[Vertex], SCollection[Edge]) = {

// First construct a data with (src, dst, name), where name can be "email", "phone", or "both"

val addressBookTypes: SCollection[((Long, Long), String)] = addressBook.flatMap { record =>

record.forwardMatches.toSeq.flatMap { matchDetails =>

val matchedUsers = (record.userId, matchDetails.userId)

(matchDetails.matchedByEmail, matchDetails.matchedByPhone) match {

case (true, true) =>

Seq((matchedUsers, EMAIL), (matchedUsers, PHONE), (matchedUsers, BOTH))

case (true, false) => Seq((matchedUsers, EMAIL))

case (false, true) => Seq((matchedUsers, PHONE))

case \_ => Seq.empty

}

}

}

// Then construct the input data for feature calculation

val addressBookFeatureInput: SCollection[InteractionGraphRawInput] = addressBookTypes

.map {

case ((src, dst), name) =>

if (src < dst)

((src, dst, name), false)

else

((dst, src, name), true)

}.groupByKey

.flatMap {

case ((src, dst, name), iterator) =>

val isReversedValues = iterator.toSeq

// check if (src, dst) is mutual follow

val isMutualFollow = isReversedValues.size == 2

// get correct srcId and dstId if there is no mutual follow and they are reversed

val (srcId, dstId) = {

if (!isMutualFollow && isReversedValues.head)

(dst, src)

else

(src, dst)

}

// get the feature name and mutual follow name

val (featureName, mfFeatureName) = name match {

case EMAIL =>

addressBookCounters.emailFeatureInc()

(FeatureName.AddressBookEmail, FeatureName.AddressBookMutualEdgeEmail)

case PHONE =>

addressBookCounters.phoneFeatureInc()

(FeatureName.AddressBookPhone, FeatureName.AddressBookMutualEdgePhone)

case BOTH =>

addressBookCounters.bothFeatureInc()

(FeatureName.AddressBookInBoth, FeatureName.AddressBookMutualEdgeInBoth)

}

// construct the TypedPipe for feature calculation

if (isMutualFollow) {

Iterator(

InteractionGraphRawInput(srcId, dstId, featureName, DefaultAge, DegaultFeatureValue),

InteractionGraphRawInput(dstId, srcId, featureName, DefaultAge, DegaultFeatureValue),

InteractionGraphRawInput(

srcId,

dstId,

mfFeatureName,

DefaultAge,

DegaultFeatureValue),

InteractionGraphRawInput(dstId, srcId, mfFeatureName, DefaultAge, DegaultFeatureValue)

)

} else {

Iterator(

InteractionGraphRawInput(srcId, dstId, featureName, DefaultAge, DegaultFeatureValue))

}

}

// Calculate the Features

FeatureGeneratorUtil.getFeatures(addressBookFeatureInput)

}

}