package com.twitter.interaction\_graph.scio.common

import com.spotify.scio.ScioMetrics

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

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

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

import com.twitter.timelines.real\_graph.v1.thriftscala.RealGraphEdgeFeatures

import com.twitter.timelines.real\_graph.v1.thriftscala.{

RealGraphEdgeFeature => RealGraphEdgeFeatureV1

}

object ConversionUtil {

def toRealGraphEdgeFeatureV1(tss: TimeSeriesStatistics): RealGraphEdgeFeatureV1 = {

RealGraphEdgeFeatureV1(

mean = Some(tss.mean),

ewma = Some(tss.ewma),

m2ForVariance = Some(tss.m2ForVariance),

daysSinceLast = tss.numDaysSinceLast.map(\_.toShort),

nonZeroDays = Some(tss.numNonZeroDays.toShort),

elapsedDays = Some(tss.numElapsedDays.toShort),

isMissing = Some(false)

)

}

/\*\*

\* Checks if the converted `RealGraphEdgeFeatures` has negative edges features.

\* Our pipeline includes other negative interactions that aren't in the UserSession thrift

\* so we'll just filter them away for now (for parity).

\*/

def hasNegativeFeatures(rgef: RealGraphEdgeFeatures): Boolean = {

rgef.numMutes.nonEmpty ||

rgef.numBlocks.nonEmpty ||

rgef.numReportAsAbuses.nonEmpty ||

rgef.numReportAsSpams.nonEmpty

}

/\*\*

\* Checks if the converted `RealGraphEdgeFeatures` has some of the key interaction features present.

\* This is adapted from timeline's code here:

\*/

def hasTimelinesRequiredFeatures(rgef: RealGraphEdgeFeatures): Boolean = {

rgef.retweetsFeature.nonEmpty ||

rgef.favsFeature.nonEmpty ||

rgef.mentionsFeature.nonEmpty ||

rgef.tweetClicksFeature.nonEmpty ||

rgef.linkClicksFeature.nonEmpty ||

rgef.profileViewsFeature.nonEmpty ||

rgef.dwellTimeFeature.nonEmpty ||

rgef.inspectedStatusesFeature.nonEmpty ||

rgef.photoTagsFeature.nonEmpty ||

rgef.numTweetQuotes.nonEmpty ||

rgef.followFeature.nonEmpty ||

rgef.mutualFollowFeature.nonEmpty ||

rgef.addressBookEmailFeature.nonEmpty ||

rgef.addressBookPhoneFeature.nonEmpty

}

/\*\*

\* Convert an Edge into a RealGraphEdgeFeature.

\* We return the converted RealGraphEdgeFeature when filterFn is true.

\* This is to allow us to filter early on during the conversion if required, rather than map over the whole

\* collection of records again to filter.

\*

\* @param filterFn true if and only if we want to keep the converted feature

\*/

def toRealGraphEdgeFeatures(

filterFn: RealGraphEdgeFeatures => Boolean

)(

e: Edge

): Option[RealGraphEdgeFeatures] = {

val baseFeature = RealGraphEdgeFeatures(destId = e.destinationId)

val aggregatedFeature = e.features.foldLeft(baseFeature) {

case (aggregatedFeature, edgeFeature) =>

val f = Some(toRealGraphEdgeFeatureV1(edgeFeature.tss))

ScioMetrics.counter("toRealGraphEdgeFeatures", edgeFeature.name.name).inc()

edgeFeature.name match {

case FeatureName.NumRetweets => aggregatedFeature.copy(retweetsFeature = f)

case FeatureName.NumFavorites => aggregatedFeature.copy(favsFeature = f)

case FeatureName.NumMentions => aggregatedFeature.copy(mentionsFeature = f)

case FeatureName.NumTweetClicks => aggregatedFeature.copy(tweetClicksFeature = f)

case FeatureName.NumLinkClicks => aggregatedFeature.copy(linkClicksFeature = f)

case FeatureName.NumProfileViews => aggregatedFeature.copy(profileViewsFeature = f)

case FeatureName.TotalDwellTime => aggregatedFeature.copy(dwellTimeFeature = f)

case FeatureName.NumInspectedStatuses =>

aggregatedFeature.copy(inspectedStatusesFeature = f)

case FeatureName.NumPhotoTags => aggregatedFeature.copy(photoTagsFeature = f)

case FeatureName.NumFollows => aggregatedFeature.copy(followFeature = f)

case FeatureName.NumMutualFollows => aggregatedFeature.copy(mutualFollowFeature = f)

case FeatureName.AddressBookEmail => aggregatedFeature.copy(addressBookEmailFeature = f)

case FeatureName.AddressBookPhone => aggregatedFeature.copy(addressBookPhoneFeature = f)

case FeatureName.AddressBookInBoth => aggregatedFeature.copy(addressBookInBothFeature = f)

case FeatureName.AddressBookMutualEdgeEmail =>

aggregatedFeature.copy(addressBookMutualEdgeEmailFeature = f)

case FeatureName.AddressBookMutualEdgePhone =>

aggregatedFeature.copy(addressBookMutualEdgePhoneFeature = f)

case FeatureName.AddressBookMutualEdgeInBoth =>

aggregatedFeature.copy(addressBookMutualEdgeInBothFeature = f)

case FeatureName.NumTweetQuotes => aggregatedFeature.copy(numTweetQuotes = f)

case FeatureName.NumBlocks => aggregatedFeature.copy(numBlocks = f)

case FeatureName.NumMutes => aggregatedFeature.copy(numMutes = f)

case FeatureName.NumReportAsSpams => aggregatedFeature.copy(numReportAsSpams = f)

case FeatureName.NumReportAsAbuses => aggregatedFeature.copy(numReportAsAbuses = f)

case \_ => aggregatedFeature

}

}

if (filterFn(aggregatedFeature))

Some(aggregatedFeature.copy(weight = e.weight.orElse(Some(0.0))))

else None

}

}