package com.twitter.interaction\_graph.scio.ml.labels

import com.spotify.scio.ScioMetrics

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

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

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

import com.twitter.interaction\_graph.thriftscala.{Edge => TEdge}

import com.twitter.socialgraph.event.thriftscala.FollowEvent

object LabelUtil {

val LabelExplicit = Set(

FeatureName.NumFollows,

FeatureName.NumFavorites,

FeatureName.NumRetweets,

FeatureName.NumMentions,

FeatureName.NumTweetQuotes,

FeatureName.NumPhotoTags,

FeatureName.NumRtFavories,

FeatureName.NumRtReplies,

FeatureName.NumRtTweetQuotes,

FeatureName.NumRtRetweets,

FeatureName.NumRtMentions,

FeatureName.NumShares,

FeatureName.NumReplies,

)

val LabelImplicit = Set(

FeatureName.NumTweetClicks,

FeatureName.NumProfileViews,

FeatureName.NumLinkClicks,

FeatureName.NumPushOpens,

FeatureName.NumNtabClicks,

FeatureName.NumRtTweetClicks,

FeatureName.NumRtLinkClicks,

FeatureName.NumEmailOpen,

FeatureName.NumEmailClick,

)

val LabelSet = (LabelExplicit ++ LabelImplicit).map(\_.value)

def fromFollowEvent(f: FollowEvent): Option[EdgeLabel] = {

for {

srcId <- f.sourceId

destId <- f.targetId

} yield EdgeLabel(srcId, destId, labels = Set(FeatureName.NumFollows))

}

def fromInteractionGraphEdge(e: TEdge): Option[EdgeLabel] = {

val labels = e.features.collect {

case EdgeFeature(featureName: FeatureName, \_) if LabelSet.contains(featureName.value) =>

ScioMetrics.counter("fromInteractionGraphEdge", featureName.toString).inc()

featureName

}.toSet

if (labels.nonEmpty) {

Some(EdgeLabel(e.sourceId, e.destinationId, labels))

} else None

}

def toTEdge(e: EdgeLabel): EdgeLabel = {

EdgeLabel(e.sourceId, e.destinationId, labels = e.labels)

}

}