package com.twitter.timelines.prediction.common.aggregates

import com.twitter.ml.api.Feature

import com.twitter.ml.api.FeatureContext

import com.twitter.ml.api.ITransform

import com.twitter.ml.api.constant.SharedFeatures

import java.lang.{Double => JDouble}

import com.twitter.timelines.prediction.common.adapters.AdapterConsumer

import com.twitter.timelines.prediction.common.adapters.EngagementLabelFeaturesDataRecordUtils

import com.twitter.ml.api.DataRecord

import com.twitter.ml.api.RichDataRecord

import com.twitter.timelines.suggests.common.engagement.thriftscala.EngagementType

import com.twitter.timelines.suggests.common.engagement.thriftscala.Engagement

import com.twitter.timelines.prediction.features.common.TimelinesSharedFeatures

import com.twitter.timelines.prediction.features.common.CombinedFeatures

/\*\*

\* To transfrom BCE events UUA data records that contain only continuous dwell time to datarecords that contain corresponding binary label features

\* The UUA datarecords inputted would have USER\_ID, SOURCE\_TWEET\_ID,TIMESTAMP and

\* 0 or one of (TWEET\_DETAIL\_DWELL\_TIME\_MS, PROFILE\_DWELL\_TIME\_MS, FULLSCREEN\_VIDEO\_DWELL\_TIME\_MS) features.

\* We will use the different engagement TIME\_MS to differentiate different engagements,

\* and then re-use the function in EngagementTypeConverte to add the binary label to the datarecord.

\*\*/

object BCELabelTransformFromUUADataRecord extends ITransform {

val dwellTimeFeatureToEngagementMap = Map(

TimelinesSharedFeatures.TWEET\_DETAIL\_DWELL\_TIME\_MS -> EngagementType.TweetDetailDwell,

TimelinesSharedFeatures.PROFILE\_DWELL\_TIME\_MS -> EngagementType.ProfileDwell,

TimelinesSharedFeatures.FULLSCREEN\_VIDEO\_DWELL\_TIME\_MS -> EngagementType.FullscreenVideoDwell

)

def dwellFeatureToEngagement(

rdr: RichDataRecord,

dwellTimeFeature: Feature[JDouble],

engagementType: EngagementType

): Option[Engagement] = {

if (rdr.hasFeature(dwellTimeFeature)) {

Some(

Engagement(

engagementType = engagementType,

timestampMs = rdr.getFeatureValue(SharedFeatures.TIMESTAMP),

weight = Some(rdr.getFeatureValue(dwellTimeFeature))

))

} else {

None

}

}

override def transformContext(featureContext: FeatureContext): FeatureContext = {

featureContext.addFeatures(

(CombinedFeatures.TweetDetailDwellEngagements ++ CombinedFeatures.ProfileDwellEngagements ++ CombinedFeatures.FullscreenVideoDwellEngagements).toSeq: \_\*)

}

override def transform(record: DataRecord): Unit = {

val rdr = new RichDataRecord(record)

val engagements = dwellTimeFeatureToEngagementMap

.map {

case (dwellTimeFeature, engagementType) =>

dwellFeatureToEngagement(rdr, dwellTimeFeature, engagementType)

}.flatten.toSeq

// Re-use BCE( behavior client events) label conversion in EngagementTypeConverter to align with BCE labels generation for offline training data

EngagementLabelFeaturesDataRecordUtils.setDwellTimeFeatures(

rdr,

Some(engagements),

AdapterConsumer.Combined)

}

}