package com.twitter.home\_mixer.product.scored\_tweets.feature\_hydrator.offline\_aggregates

import com.twitter.finagle.stats.NullStatsReceiver

import com.twitter.timelinemixer.injection.repository.uss.VersionedAggregateFeaturesDecoder

import com.twitter.ml.api.DataRecord

import com.twitter.timelines.aggregate\_interactions.thriftjava.UserAggregateInteractions

import com.twitter.timelines.aggregate\_interactions.v17.thriftjava.{

UserAggregateInteractions => V17UserAggregateInteractions

}

import com.twitter.timelines.aggregate\_interactions.v1.thriftjava.{

UserAggregateInteractions => V1UserAggregateInteractions

}

import com.twitter.timelines.suggests.common.dense\_data\_record.thriftjava.DenseCompactDataRecord

import com.twitter.timelines.suggests.common.dense\_data\_record.thriftscala.DenseFeatureMetadata

import java.lang.{Long => JLong}

import java.util.Collections

import java.util.{Map => JMap}

private[offline\_aggregates] case class AggregateFeaturesToDecodeWithMetadata(

metadataOpt: Option[DenseFeatureMetadata],

aggregates: UserAggregateInteractions) {

def toDataRecord(dr: DenseCompactDataRecord): DataRecord =

VersionedAggregateFeaturesDecoder.fromJDenseCompact(

metadataOpt,

dr.versionId,

NullStatsReceiver,

s"V${dr.versionId}"

)(dr)

def userAggregatesOpt: Option[DenseCompactDataRecord] = {

aggregates.getSetField match {

case UserAggregateInteractions.\_Fields.V17 =>

Option(aggregates.getV17.user\_aggregates)

case \_ =>

None

}

}

def userAuthorAggregates = extract(\_.user\_author\_aggregates)

def userEngagerAggregates = extract(\_.user\_engager\_aggregates)

def userMentionAggregates = extract(\_.user\_mention\_aggregates)

def userOriginalAuthorAggregates = extract(\_.user\_original\_author\_aggregates)

def userRequestDowAggregates = extract(\_.user\_request\_dow\_aggregates)

def userRequestHourAggregates = extract(\_.user\_request\_hour\_aggregates)

def rectweetUserSimclustersTweetAggregates = extract(\_.rectweet\_user\_simclusters\_tweet\_aggregates)

def userTwitterListAggregates = extract(\_.user\_list\_aggregates)

def userTopicAggregates = extract(\_.user\_topic\_aggregates)

def userInferredTopicAggregates = extract(\_.user\_inferred\_topic\_aggregates)

def userMediaUnderstandingAnnotationAggregates = extract(

\_.user\_media\_understanding\_annotation\_aggregates)

private def extract[T](

v17Fn: V17UserAggregateInteractions => JMap[JLong, DenseCompactDataRecord]

): JMap[JLong, DenseCompactDataRecord] = {

aggregates.getSetField match {

case UserAggregateInteractions.\_Fields.V17 =>

v17Fn(aggregates.getV17)

case \_ =>

Collections.emptyMap()

}

}

}

object AggregateFeaturesToDecodeWithMetadata {

val empty = new AggregateFeaturesToDecodeWithMetadata(

None,

UserAggregateInteractions.v1(new V1UserAggregateInteractions()))

}