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

import com.twitter.ml.api.DataRecord

import com.twitter.ml.api.FeatureContext

import com.twitter.ml.api.RichDataRecord

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

private[offline\_aggregates] object Utils {

/\*\*

\* Selects only those values in map that correspond to the keys in ids and apply the provided

\* transform to the selected values. This is a convenience method for use by Timelines Aggregation

\* Framework based features.

\*

\* @param idsToSelect The set of ids to extract values for.

\* @param transform A transform to apply to the selected values.

\* @param map Map[Long, DenseCompactDataRecord]

\*/

def selectAndTransform(

idsToSelect: Seq[Long],

transform: DenseCompactDataRecord => DataRecord,

map: java.util.Map[java.lang.Long, DenseCompactDataRecord],

): Map[Long, DataRecord] = {

val filtered: Seq[(Long, DataRecord)] =

for {

id <- idsToSelect if map.containsKey(id)

} yield {

id -> transform(map.get(id))

}

filtered.toMap

}

def filterDataRecord(dr: DataRecord, featureContext: FeatureContext): Unit = {

new RichDataRecord(dr, featureContext).dropUnknownFeatures()

}

}