package com.twitter.timelines.data\_processing.ml\_util.aggregation\_framework.metrics

import com.twitter.ml.api.\_

import com.twitter.util.Duration

import com.twitter.util.Time

import java.lang.{Double => JDouble}

import java.lang.{Long => JLong}

import java.util.{Map => JMap}

/\*

\* TypedSumLikeMetric aggregates a sum over any feature transform.

\* TypedCountMetric, TypedSumMetric, TypedSumSqMetric are examples

\* of metrics that are inherited from this trait. To implement a new

\* "sum like" metric, override the getIncrementValue() and operatorName

\* members of this trait.

\*

\* getIncrementValue() is inherited from the

\* parent trait AggregationMetric, but not overriden in this trait, so

\* it needs to be overloaded by any metric that extends TypedSumLikeMetric.

\*

\* operatorName is a string used for naming the resultant aggregate feature

\* (e.g. "count" if its a count feature, or "sum" if a sum feature).

\*/

trait TypedSumLikeMetric[T] extends TimedValueAggregationMetric[T] {

import AggregationMetricCommon.\_

def useFixedDecay = true

override def plus(

left: TimedValue[Double],

right: TimedValue[Double],

halfLife: Duration

): TimedValue[Double] = {

val resultValue = if (halfLife == Duration.Top) {

/\* We could use decayedValueMonoid here, but

\* a simple addition is slightly more accurate \*/

left.value + right.value

} else {

val decayedLeft = toDecayedValue(left, halfLife)

val decayedRight = toDecayedValue(right, halfLife)

decayedValueMonoid.plus(decayedLeft, decayedRight).value

}

TimedValue[Double](

resultValue,

left.timestamp.max(right.timestamp)

)

}

override def zero(timeOpt: Option[Long]): TimedValue[Double] = {

val timestamp =

/\*

\* Please see TQ-11279 for documentation for this fix to the decay logic.

\*/

if (useFixedDecay) {

Time.fromMilliseconds(timeOpt.getOrElse(0L))

} else {

Time.fromMilliseconds(0L)

}

TimedValue[Double](

value = 0.0,

timestamp = timestamp

)

}

}