package com.twitter.simclusters\_v2.summingbird.common

import com.twitter.algebird.{Monoid, OptionMonoid}

import com.twitter.simclusters\_v2.common.SimClustersEmbedding

import com.twitter.simclusters\_v2.summingbird.common.Monoids.TopKScoresUtils

import com.twitter.simclusters\_v2.thriftscala.{

SimClustersEmbeddingMetadata,

SimClustersEmbeddingWithMetadata,

SimClustersEmbedding => ThriftSimClustersEmbedding

}

/\*\*

\* Decayed aggregation of embeddings.

\*

\* When merging 2 embeddings, the older embedding's scores are scaled by time. If a cluster is

\* present in both embeddings, the highest score (after scaling) is used in the result.

\*

\* @halfLifeMs - defines how quickly a score decays

\* @topK - only the topk clusters with the highest scores are retained in the result

\* @threshold - any clusters with weights below threshold are excluded from the result

\*/

class SimClustersEmbeddingWithMetadataMonoid(

halfLifeMs: Long,

topK: Int,

threshold: Double)

extends Monoid[SimClustersEmbeddingWithMetadata] {

override val zero: SimClustersEmbeddingWithMetadata = SimClustersEmbeddingWithMetadata(

ThriftSimClustersEmbedding(),

SimClustersEmbeddingMetadata()

)

private val optionLongMonoid = new OptionMonoid[Long]()

private val optionMaxMonoid =

new OptionMonoid[Long]()(com.twitter.algebird.Max.maxSemigroup[Long])

override def plus(

x: SimClustersEmbeddingWithMetadata,

y: SimClustersEmbeddingWithMetadata

): SimClustersEmbeddingWithMetadata = {

val mergedClusterScores = TopKScoresUtils.mergeClusterScoresWithUpdateTimes(

x = SimClustersEmbedding(x.embedding).embedding,

xUpdatedAtMs = x.metadata.updatedAtMs.getOrElse(0),

y = SimClustersEmbedding(y.embedding).embedding,

yUpdatedAtMs = y.metadata.updatedAtMs.getOrElse(0),

halfLifeMs = halfLifeMs,

topK = topK,

threshold = threshold

)

SimClustersEmbeddingWithMetadata(

embedding = SimClustersEmbedding(mergedClusterScores).toThrift,

metadata = SimClustersEmbeddingMetadata(

updatedAtMs = optionMaxMonoid.plus(x.metadata.updatedAtMs, y.metadata.updatedAtMs),

updatedCount = optionLongMonoid.plus(x.metadata.updatedCount, y.metadata.updatedCount)

)

)

}

}