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

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

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

class MedoidRepresentativeSelectionMethod[T](

producerProducerSimilarityFn: (T, T) => Double)

extends ClusterRepresentativeSelectionMethod[T] {

/\*\*

\* Identify the medoid of a cluster and return it.

\*

\* @param cluster A set of NeighborWithWeights.

\* @param embeddings A map of producer ID -> embedding.

\*/

def selectClusterRepresentative(

cluster: Set[NeighborWithWeights],

embeddings: Map[UserId, T],

): UserId = {

val key = cluster.maxBy {

id1 => // maxBy because we use similarity, which gets larger as we get closer.

val v = embeddings(id1.neighborId)

cluster

.map(id2 => producerProducerSimilarityFn(v, embeddings(id2.neighborId))).sum

}

key.neighborId

}

}