package com.twitter.simclusters\_v2.stores

import com.twitter.bijection.Bufferable

import com.twitter.bijection.Injection

import com.twitter.bijection.scrooge.CompactScalaCodec

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

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

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

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

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

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

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

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

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

import com.twitter.storage.client.manhattan.kv.ManhattanKVClientMtlsParams

import com.twitter.storehaus.ReadableStore

import com.twitter.storehaus\_internal.manhattan.Apollo

import com.twitter.storehaus\_internal.manhattan.ManhattanRO

import com.twitter.storehaus\_internal.manhattan.ManhattanROConfig

import com.twitter.storehaus\_internal.util.ApplicationID

import com.twitter.storehaus\_internal.util.DatasetName

import com.twitter.storehaus\_internal.util.HDFSPath

import com.twitter.scalding\_internal.multiformat.format.keyval.KeyValInjection.Long2BigEndian

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

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

object MultiTypeGraphStore {

implicit val leftNodesInject: Injection[LeftNode, Array[Byte]] =

CompactScalaCodec(LeftNode)

implicit val truncatedMultiTypeGraphInject: Injection[RightNodeWithEdgeWeightList, Array[Byte]] =

CompactScalaCodec(RightNodeWithEdgeWeightList)

implicit val topKNounsListInject: Injection[NounWithFrequencyList, Array[Byte]] =

CompactScalaCodec(NounWithFrequencyList)

implicit val rightNodesStructInject: Injection[RightNodeTypeStruct, Array[Byte]] =

CompactScalaCodec(RightNodeTypeStruct)

implicit val similarRightNodesStructInject: Injection[SimilarRightNodes, Array[Byte]] =

CompactScalaCodec(SimilarRightNodes)

implicit val rightNodesInject: Injection[RightNode, Array[Byte]] =

CompactScalaCodec(RightNode)

implicit val tweetCandidatesInject: Injection[CandidateTweetsList, Array[Byte]] =

CompactScalaCodec(CandidateTweetsList)

implicit val fullClusterIdInject: Injection[FullClusterId, Array[Byte]] =

CompactScalaCodec(FullClusterId)

implicit val topKTweetsWithScoresInject: Injection[TopKTweetsWithScores, Array[Byte]] =

CompactScalaCodec(TopKTweetsWithScores)

implicit val clustersUserIsInterestedInInjection: Injection[ClustersUserIsInterestedIn, Array[

Byte

]] =

CompactScalaCodec(ClustersUserIsInterestedIn)

private val appId = "multi\_type\_simclusters"

def getTruncatedMultiTypeGraphRightNodesForUser(

mhMtlsParams: ManhattanKVClientMtlsParams

): ReadableStore[LeftNode, RightNodeWithEdgeWeightList] = {

ManhattanRO.getReadableStoreWithMtls[LeftNode, RightNodeWithEdgeWeightList](

ManhattanROConfig(

HDFSPath(""),

ApplicationID(appId),

DatasetName("mts\_user\_truncated\_graph"),

Apollo

),

mhMtlsParams

)

}

def getTopKNounsForRightNodeType(

mhMtlsParams: ManhattanKVClientMtlsParams

): ReadableStore[RightNodeTypeStruct, NounWithFrequencyList] = {

ManhattanRO.getReadableStoreWithMtls[RightNodeTypeStruct, NounWithFrequencyList](

ManhattanROConfig(

HDFSPath(""),

ApplicationID(appId),

DatasetName("mts\_topk\_frequent\_nouns"),

Apollo

),

mhMtlsParams

)

}

def getTopKSimilarRightNodes(

mhMtlsParams: ManhattanKVClientMtlsParams

): ReadableStore[RightNode, SimilarRightNodes] = {

ManhattanRO.getReadableStoreWithMtls[RightNode, SimilarRightNodes](

ManhattanROConfig(

HDFSPath(""),

ApplicationID(appId),

DatasetName("mts\_topk\_similar\_right\_nodes\_scio"),

Apollo

),

mhMtlsParams

)

}

def getOfflineTweetMTSCandidateStore(

mhMtlsParams: ManhattanKVClientMtlsParams

): ReadableStore[Long, CandidateTweetsList] = {

ManhattanRO.getReadableStoreWithMtls[Long, CandidateTweetsList](

ManhattanROConfig(

HDFSPath(""),

ApplicationID(appId),

DatasetName("offline\_tweet\_recommendations\_from\_mts\_consumer\_embeddings"),

Apollo

),

mhMtlsParams

)

}

def getOfflineTweet2020CandidateStore(

mhMtlsParams: ManhattanKVClientMtlsParams

): ReadableStore[Long, CandidateTweetsList] = {

ManhattanRO.getReadableStoreWithMtls[Long, CandidateTweetsList](

ManhattanROConfig(

HDFSPath(""),

ApplicationID(appId),

DatasetName("offline\_tweet\_recommendations\_from\_interestedin\_2020"),

Apollo

),

mhMtlsParams

)

}

def getVideoViewBasedClusterTopKTweets(

mhMtlsParams: ManhattanKVClientMtlsParams

): ReadableStore[FullClusterId, TopKTweetsWithScores] = {

ManhattanRO

.getReadableStoreWithMtls[FullClusterId, TopKTweetsWithScores](

ManhattanROConfig(

HDFSPath(""),

ApplicationID(appId),

DatasetName("video\_view\_based\_cluster\_to\_tweet\_index"),

Apollo

),

mhMtlsParams

)

}

def getRetweetBasedClusterTopKTweets(

mhMtlsParams: ManhattanKVClientMtlsParams

): ReadableStore[FullClusterId, TopKTweetsWithScores] = {

ManhattanRO

.getReadableStoreWithMtls[FullClusterId, TopKTweetsWithScores](

ManhattanROConfig(

HDFSPath(""),

ApplicationID(appId),

DatasetName("retweet\_based\_simclusters\_cluster\_to\_tweet\_index"),

Apollo

),

mhMtlsParams

)

}

def getReplyBasedClusterTopKTweets(

mhMtlsParams: ManhattanKVClientMtlsParams

): ReadableStore[FullClusterId, TopKTweetsWithScores] = {

ManhattanRO

.getReadableStoreWithMtls[FullClusterId, TopKTweetsWithScores](

ManhattanROConfig(

HDFSPath(""),

ApplicationID(appId),

DatasetName("reply\_based\_simclusters\_cluster\_to\_tweet\_index"),

Apollo

),

mhMtlsParams

)

}

def getPushOpenBasedClusterTopKTweets(

mhMtlsParams: ManhattanKVClientMtlsParams

): ReadableStore[FullClusterId, TopKTweetsWithScores] = {

ManhattanRO

.getReadableStoreWithMtls[FullClusterId, TopKTweetsWithScores](

ManhattanROConfig(

HDFSPath(""),

ApplicationID(appId),

DatasetName("push\_open\_based\_simclusters\_cluster\_to\_tweet\_index"),

Apollo

),

mhMtlsParams

)

}

def getAdsFavBasedClusterTopKTweets(

mhMtlsParams: ManhattanKVClientMtlsParams

): ReadableStore[FullClusterId, TopKTweetsWithScores] = {

ManhattanRO

.getReadableStoreWithMtls[FullClusterId, TopKTweetsWithScores](

ManhattanROConfig(

HDFSPath(""),

ApplicationID(appId),

DatasetName("ads\_fav\_based\_simclusters\_cluster\_to\_tweet\_index"),

Apollo

),

mhMtlsParams

)

}

def getAdsFavClickBasedClusterTopKTweets(

mhMtlsParams: ManhattanKVClientMtlsParams

): ReadableStore[FullClusterId, TopKTweetsWithScores] = {

ManhattanRO

.getReadableStoreWithMtls[FullClusterId, TopKTweetsWithScores](

ManhattanROConfig(

HDFSPath(""),

ApplicationID(appId),

DatasetName("ads\_fav\_click\_based\_simclusters\_cluster\_to\_tweet\_index"),

Apollo

),

mhMtlsParams

)

}

def getFTRPop1000BasedClusterTopKTweets(

mhMtlsParams: ManhattanKVClientMtlsParams

): ReadableStore[FullClusterId, TopKTweetsWithScores] = {

ManhattanRO

.getReadableStoreWithMtls[FullClusterId, TopKTweetsWithScores](

ManhattanROConfig(

HDFSPath(""),

ApplicationID(appId),

DatasetName("ftr\_pop1000\_rank\_decay\_1\_1\_cluster\_to\_tweet\_index"),

Apollo

),

mhMtlsParams

)

}

def getFTRPop10000BasedClusterTopKTweets(

mhMtlsParams: ManhattanKVClientMtlsParams

): ReadableStore[FullClusterId, TopKTweetsWithScores] = {

ManhattanRO

.getReadableStoreWithMtls[FullClusterId, TopKTweetsWithScores](

ManhattanROConfig(

HDFSPath(""),

ApplicationID(appId),

DatasetName("ftr\_pop10000\_rank\_decay\_1\_1\_cluster\_to\_tweet\_index"),

Apollo

),

mhMtlsParams

)

}

def getOONFTRPop1000BasedClusterTopKTweets(

mhMtlsParams: ManhattanKVClientMtlsParams

): ReadableStore[FullClusterId, TopKTweetsWithScores] = {

ManhattanRO

.getReadableStoreWithMtls[FullClusterId, TopKTweetsWithScores](

ManhattanROConfig(

HDFSPath(""),

ApplicationID(appId),

DatasetName("oon\_ftr\_pop1000\_rnkdecay\_cluster\_to\_tweet\_index"),

Apollo

),

mhMtlsParams

)

}

def getOfflineLogFavBasedTweetBasedClusterTopKTweets(

mhMtlsParams: ManhattanKVClientMtlsParams

): ReadableStore[FullClusterId, TopKTweetsWithScores] = {

ManhattanRO

.getReadableStoreWithMtls[FullClusterId, TopKTweetsWithScores](

ManhattanROConfig(

HDFSPath(""),

ApplicationID(appId),

DatasetName("decayed\_sum\_cluster\_to\_tweet\_index"),

Apollo

),

mhMtlsParams

)

}

def getGlobalSimClustersLanguageEmbeddings(

mhMtlsParams: ManhattanKVClientMtlsParams

): ReadableStore[Language, ClustersUserIsInterestedIn] = {

ManhattanRO

.getReadableStoreWithMtls[Language, ClustersUserIsInterestedIn](

ManhattanROConfig(

HDFSPath(""),

ApplicationID(appId),

DatasetName("global\_simclusters\_language\_embeddings"),

Apollo

),

mhMtlsParams

)

}

}