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

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.frigate.common.store.strato.StratoFetchableStore

import com.twitter.frigate.common.store.strato.StratoStore

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

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

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

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

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

import com.twitter.storehaus.ReadableStore

import com.twitter.storehaus.Store

import com.twitter.strato.catalog.Scan.Slice

import com.twitter.strato.client.Client

import com.twitter.strato.thrift.ScroogeConvImplicits.\_

object PersistentTweetEmbeddingStore {

val LogFavBasedColumn =

"recommendations/simclusters\_v2/embeddings/logFavBasedTweet20M145KUpdatedPersistent"

val LogFavBasedColumn20m145k2020 =

"recommendations/simclusters\_v2/embeddings/logFavBasedTweet20M145K2020Persistent"

val LogFavBased20m145k2020Dataset = "log\_fav\_based\_tweet\_20m\_145k\_2020\_embeddings"

val LogFavBased20m145kUpdatedDataset = "log\_fav\_based\_tweet\_20m\_145k\_updated\_embeddings"

val DefaultMaxLength = 15

def mostRecentTweetEmbeddingStore(

stratoClient: Client,

column: String,

maxLength: Int = DefaultMaxLength

): ReadableStore[TweetId, SimClustersEmbedding] = {

StratoFetchableStore

.withUnitView[(TweetId, Timestamp), PersistentSimClustersEmbedding](stratoClient, column)

.composeKeyMapping[TweetId]((\_, LatestEmbeddingVersion))

.mapValues(\_.embedding.truncate(maxLength))

}

def longestL2NormTweetEmbeddingStore(

stratoClient: Client,

column: String

): ReadableStore[TweetId, SimClustersEmbedding] =

StratoFetchableStore

.withUnitView[(TweetId, Timestamp), PersistentSimClustersEmbedding](stratoClient, column)

.composeKeyMapping[TweetId]((\_, LongestL2EmbeddingVersion))

.mapValues(\_.embedding)

def mostRecentTweetEmbeddingStoreManhattan(

mhMtlsParams: ManhattanKVClientMtlsParams,

dataset: String,

statsReceiver: StatsReceiver,

maxLength: Int = DefaultMaxLength

): ReadableStore[TweetId, SimClustersEmbedding] =

ManhattanFromStratoStore

.createPersistentTweetStore(

dataset = dataset,

mhMtlsParams = mhMtlsParams,

statsReceiver = statsReceiver

).composeKeyMapping[TweetId]((\_, LatestEmbeddingVersion))

.mapValues[SimClustersEmbedding](\_.embedding.truncate(maxLength))

def longestL2NormTweetEmbeddingStoreManhattan(

mhMtlsParams: ManhattanKVClientMtlsParams,

dataset: String,

statsReceiver: StatsReceiver,

maxLength: Int = 50

): ReadableStore[TweetId, SimClustersEmbedding] =

ManhattanFromStratoStore

.createPersistentTweetStore(

dataset = dataset,

mhMtlsParams = mhMtlsParams,

statsReceiver = statsReceiver

).composeKeyMapping[TweetId]((\_, LongestL2EmbeddingVersion))

.mapValues[SimClustersEmbedding](\_.embedding.truncate(maxLength))

/\*\*

\* The writeable store for Persistent Tweet embedding. Only available in SimClusters package.

\*/

private[simclusters\_v2] def persistentTweetEmbeddingStore(

stratoClient: Client,

column: String

): Store[PersistentTweetEmbeddingId, PersistentSimClustersEmbedding] = {

StratoStore

.withUnitView[(TweetId, Timestamp), PersistentSimClustersEmbedding](stratoClient, column)

.composeKeyMapping(\_.toTuple)

}

type Timestamp = Long

case class PersistentTweetEmbeddingId(

tweetId: TweetId,

timestampInMs: Timestamp = LatestEmbeddingVersion) {

lazy val toTuple: (TweetId, Timestamp) = (tweetId, timestampInMs)

}

// Special version - reserved for the latest version of the embedding

private[summingbird] val LatestEmbeddingVersion = 0L

// Special version - reserved for the embedding with the longest L2 norm

private[summingbird] val LongestL2EmbeddingVersion = 1L

// The tweet embedding store keeps at most 20 LKeys

private[stores] val DefaultSlice = Slice[Long](from = None, to = None, limit = None)

}