package com.twitter.representationscorer.scorestore

import com.twitter.bijection.scrooge.BinaryScalaCodec

import com.twitter.conversions.DurationOps.\_

import com.twitter.finagle.memcached.Client

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.hashing.KeyHasher

import com.twitter.hermit.store.common.ObservedCachedReadableStore

import com.twitter.hermit.store.common.ObservedMemcachedReadableStore

import com.twitter.hermit.store.common.ObservedReadableStore

import com.twitter.relevance\_platform.common.injection.LZ4Injection

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

import com.twitter.simclusters\_v2.score.ScoreFacadeStore

import com.twitter.simclusters\_v2.score.SimClustersEmbeddingPairScoreStore

import com.twitter.simclusters\_v2.thriftscala.EmbeddingType.FavTfgTopic

import com.twitter.simclusters\_v2.thriftscala.EmbeddingType.LogFavBasedKgoApeTopic

import com.twitter.simclusters\_v2.thriftscala.EmbeddingType.LogFavBasedTweet

import com.twitter.simclusters\_v2.thriftscala.ModelVersion.Model20m145kUpdated

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

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

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

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

import com.twitter.stitch.storehaus.StitchOfReadableStore

import com.twitter.storehaus.ReadableStore

import com.twitter.strato.client.{Client => StratoClient}

import com.twitter.topic\_recos.stores.CertoTweetTopicScoresStore

import javax.inject.Inject

import javax.inject.Singleton

@Singleton()

class ScoreStore @Inject() (

simClustersEmbeddingStore: ReadableStore[SimClustersEmbeddingId, SimClustersEmbedding],

stratoClient: StratoClient,

representationScorerCacheClient: Client,

stats: StatsReceiver) {

private val keyHasher = KeyHasher.FNV1A\_64

private val statsReceiver = stats.scope("score\_store")

/\*\* \*\* Score Store \*\*\*\*\*/

private val simClustersEmbeddingCosineSimilarityScoreStore =

ObservedReadableStore(

SimClustersEmbeddingPairScoreStore

.buildCosineSimilarityStore(simClustersEmbeddingStore)

.toThriftStore

)(statsReceiver.scope("simClusters\_embedding\_cosine\_similarity\_score\_store"))

private val simClustersEmbeddingDotProductScoreStore =

ObservedReadableStore(

SimClustersEmbeddingPairScoreStore

.buildDotProductStore(simClustersEmbeddingStore)

.toThriftStore

)(statsReceiver.scope("simClusters\_embedding\_dot\_product\_score\_store"))

private val simClustersEmbeddingJaccardSimilarityScoreStore =

ObservedReadableStore(

SimClustersEmbeddingPairScoreStore

.buildJaccardSimilarityStore(simClustersEmbeddingStore)

.toThriftStore

)(statsReceiver.scope("simClusters\_embedding\_jaccard\_similarity\_score\_store"))

private val simClustersEmbeddingEuclideanDistanceScoreStore =

ObservedReadableStore(

SimClustersEmbeddingPairScoreStore

.buildEuclideanDistanceStore(simClustersEmbeddingStore)

.toThriftStore

)(statsReceiver.scope("simClusters\_embedding\_euclidean\_distance\_score\_store"))

private val simClustersEmbeddingManhattanDistanceScoreStore =

ObservedReadableStore(

SimClustersEmbeddingPairScoreStore

.buildManhattanDistanceStore(simClustersEmbeddingStore)

.toThriftStore

)(statsReceiver.scope("simClusters\_embedding\_manhattan\_distance\_score\_store"))

private val simClustersEmbeddingLogCosineSimilarityScoreStore =

ObservedReadableStore(

SimClustersEmbeddingPairScoreStore

.buildLogCosineSimilarityStore(simClustersEmbeddingStore)

.toThriftStore

)(statsReceiver.scope("simClusters\_embedding\_log\_cosine\_similarity\_score\_store"))

private val simClustersEmbeddingExpScaledCosineSimilarityScoreStore =

ObservedReadableStore(

SimClustersEmbeddingPairScoreStore

.buildExpScaledCosineSimilarityStore(simClustersEmbeddingStore)

.toThriftStore

)(statsReceiver.scope("simClusters\_embedding\_exp\_scaled\_cosine\_similarity\_score\_store"))

// Use the default setting

private val topicTweetRankingScoreStore =

TopicTweetRankingScoreStore.buildTopicTweetRankingStore(

FavTfgTopic,

LogFavBasedKgoApeTopic,

LogFavBasedTweet,

Model20m145kUpdated,

consumerEmbeddingMultiplier = 1.0,

producerEmbeddingMultiplier = 1.0

)

private val topicTweetsCortexThresholdStore = TopicTweetsCosineSimilarityAggregateStore(

TopicTweetsCosineSimilarityAggregateStore.DefaultScoreKeys,

statsReceiver.scope("topic\_tweets\_cortex\_threshold\_store")

)

val topicTweetCertoScoreStore: ObservedCachedReadableStore[ScoreId, Score] = {

val underlyingStore = ObservedReadableStore(

TopicTweetCertoScoreStore(CertoTweetTopicScoresStore.prodStore(stratoClient))

)(statsReceiver.scope("topic\_tweet\_certo\_score\_store"))

val memcachedStore = ObservedMemcachedReadableStore

.fromCacheClient(

backingStore = underlyingStore,

cacheClient = representationScorerCacheClient,

ttl = 10.minutes

)(

valueInjection = LZ4Injection.compose(BinaryScalaCodec(Score)),

statsReceiver = statsReceiver.scope("topic\_tweet\_certo\_store\_memcache"),

keyToString = { k: ScoreId =>

s"certocs:${keyHasher.hashKey(k.toString.getBytes)}"

}

)

ObservedCachedReadableStore.from[ScoreId, Score](

memcachedStore,

ttl = 5.minutes,

maxKeys = 1000000,

cacheName = "topic\_tweet\_certo\_store\_cache",

windowSize = 10000L

)(statsReceiver.scope("topic\_tweet\_certo\_store\_cache"))

}

val uniformScoringStore: ReadableStore[ScoreId, Score] =

ScoreFacadeStore.buildWithMetrics(

readableStores = Map(

ScoringAlgorithm.PairEmbeddingCosineSimilarity ->

simClustersEmbeddingCosineSimilarityScoreStore,

ScoringAlgorithm.PairEmbeddingDotProduct ->

simClustersEmbeddingDotProductScoreStore,

ScoringAlgorithm.PairEmbeddingJaccardSimilarity ->

simClustersEmbeddingJaccardSimilarityScoreStore,

ScoringAlgorithm.PairEmbeddingEuclideanDistance ->

simClustersEmbeddingEuclideanDistanceScoreStore,

ScoringAlgorithm.PairEmbeddingManhattanDistance ->

simClustersEmbeddingManhattanDistanceScoreStore,

ScoringAlgorithm.PairEmbeddingLogCosineSimilarity ->

simClustersEmbeddingLogCosineSimilarityScoreStore,

ScoringAlgorithm.PairEmbeddingExpScaledCosineSimilarity ->

simClustersEmbeddingExpScaledCosineSimilarityScoreStore,

// Certo normalized cosine score between topic-tweet pairs

ScoringAlgorithm.CertoNormalizedCosineScore

-> topicTweetCertoScoreStore,

// Certo normalized dot-product score between topic-tweet pairs

ScoringAlgorithm.CertoNormalizedDotProductScore

-> topicTweetCertoScoreStore

),

aggregatedStores = Map(

ScoringAlgorithm.WeightedSumTopicTweetRanking ->

topicTweetRankingScoreStore,

ScoringAlgorithm.CortexTopicTweetLabel ->

topicTweetsCortexThresholdStore,

),

statsReceiver = stats

)

val uniformScoringStoreStitch: ScoreId => com.twitter.stitch.Stitch[Score] =

StitchOfReadableStore(uniformScoringStore)

}