package com.twitter.simclusters\_v2.scalding.topic\_recommendations.model\_based\_topic\_recommendations

import com.twitter.scalding.{DateRange, Days, Stat, TypedPipe, UniqueID}

import com.twitter.scalding\_internal.dalv2.DAL

import com.twitter.scalding\_internal.dalv2.remote\_access.{ExplicitLocation, Proc3Atla}

import com.twitter.scalding\_internal.multiformat.format.keyval.KeyVal

import com.twitter.simclusters\_v2.common.{Language, TopicId, UserId}

import com.twitter.simclusters\_v2.hdfs\_sources.FavTfgTopicEmbeddingsScalaDataset

import com.twitter.simclusters\_v2.scalding.embedding.common.ExternalDataSources

import com.twitter.simclusters\_v2.summingbird.stores.UserInterestedInReadableStore

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

EmbeddingType,

InternalId,

LocaleEntityId,

ModelVersion,

SimClustersEmbeddingId

}

import java.util.TimeZone

/\*\*

\* DataSources object to read datasets for the model based topic recommendations

\*/

object DataSources {

private val topicEmbeddingDataset = FavTfgTopicEmbeddingsScalaDataset

private val topicEmbeddingType = EmbeddingType.FavTfgTopic

/\*\*

\* Get user InterestedIn data, filter popular clusters and return fav-scores interestedIn embedding for user

\*/

def getUserInterestedInData(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueID: UniqueID

): TypedPipe[(UserId, Map[Int, Double])] = {

val numUserInterestedInInput = Stat("num\_user\_interested\_in")

ExternalDataSources.simClustersInterestInSource

.map {

case KeyVal(userId, clustersUserIsInterestedIn) =>

val clustersPostFiltering = clustersUserIsInterestedIn.clusterIdToScores.filter {

case (clusterId, clusterScores) =>

// filter out popular clusters (i.e clusters with > 5M users interested in it) from the user embedding

clusterScores.numUsersInterestedInThisClusterUpperBound.exists(

\_ < UserInterestedInReadableStore.MaxClusterSizeForUserInterestedInDataset)

}

numUserInterestedInInput.inc()

(userId, clustersPostFiltering.mapValues(\_.favScore.getOrElse(0.0)).toMap)

}

}

def getPerLanguageTopicEmbeddings(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueID: UniqueID

): TypedPipe[((TopicId, Language), Map[Int, Double])] = {

val numTFGPerLanguageEmbeddings = Stat("num\_per\_language\_tfg\_embeddings")

DAL

.readMostRecentSnapshotNoOlderThan(topicEmbeddingDataset, Days(30))

.withRemoteReadPolicy(ExplicitLocation(Proc3Atla))

.toTypedPipe

.map {

case KeyVal(k, v) => (k, v)

}.collect {

case (

SimClustersEmbeddingId(

embedType,

ModelVersion.Model20m145kUpdated,

InternalId.LocaleEntityId(LocaleEntityId(entityId, lang))),

embedding) if (embedType == topicEmbeddingType) =>

numTFGPerLanguageEmbeddings.inc()

((entityId, lang), embedding.embedding.map(\_.toTuple).toMap)

}.forceToDisk

}

}