package com.twitter.simclusters\_v2.scalding.topic\_recommendations

import com.twitter.bijection.Bufferable

import com.twitter.bijection.Injection

import com.twitter.recos.entities.thriftscala.SemanticCoreEntity

import com.twitter.recos.entities.thriftscala.SemanticCoreEntityScoreList

import com.twitter.recos.entities.thriftscala.SemanticEntityScore

import com.twitter.scalding.commons.source.VersionedKeyValSource

import com.twitter.scalding.Execution

import com.twitter.scalding.\_

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

import com.twitter.scalding\_internal.dalv2.DALWrite.\_

import com.twitter.scalding\_internal.dalv2.remote\_access.ExplicitLocation

import com.twitter.scalding\_internal.dalv2.remote\_access.Proc2Atla

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

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

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

import com.twitter.timelines.per\_topic\_metrics.thriftscala.PerTopicAggregateEngagementMetric

import com.twitter.wtf.scalding.jobs.common.AdhocExecutionApp

import com.twitter.wtf.scalding.jobs.common.ScheduledExecutionApp

import java.util.TimeZone

import timelines.data\_processing.jobs.metrics.per\_topic\_metrics.PerTopicAggregateEngagementScalaDataset

/\*\*

scalding remote run \

--target src/scala/com/twitter/simclusters\_v2/scalding/topic\_recommendations:geopopular\_top\_tweets\_impressed\_topics\_adhoc \

--main-class com.twitter.simclusters\_v2.scalding.topic\_recommendations.GeoPopularTopicsAdhocApp \

--submitter hadoopnest1.atla.twitter.com --user recos-platform \

-- \

--date 2020-03-28 --output\_dir /user/recos-platform/adhoc/your\_ldap/topics\_country\_counts

\*/

object GeoPopularTopicsAdhocApp extends AdhocExecutionApp {

override def runOnDateRange(

args: Args

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueID: UniqueID

): Execution[Unit] = {

val maxTopicsPerCountry = args.int("maxTopics", 2000)

val typedTsv = args.boolean("tsv")

implicit val inj: Injection[List[(SemanticCoreEntityId, Double)], Array[Byte]] =

Bufferable.injectionOf[List[(SemanticCoreEntityId, Double)]]

val perTopicEngagementLogData = DAL

.read(PerTopicAggregateEngagementScalaDataset, dateRange.prepend(Days(7)))

.toTypedPipe

val topicsWithEngagement =

GeoPopularTopicsApp

.getPopularTopicsFromLogs(perTopicEngagementLogData, maxTopicsPerCountry)

.mapValues(\_.toList)

if (typedTsv) {

topicsWithEngagement.writeExecution(

TypedTsv(args("/user/recos-platform/adhoc/your\_ldap/topics\_country\_counts\_tsv"))

)

} else {

topicsWithEngagement.writeExecution(

VersionedKeyValSource[String, List[(SemanticCoreEntityId, Double)]](args("output\_dir"))

)

}

}

}

/\*\*

capesospy-v2 update --build\_locally \

--start\_cron popular\_topics\_per\_country \

src/scala/com/twitter/simclusters\_v2/capesos\_config/atla\_proc3.yaml

\*/

object GeoPopularTopicsBatchApp extends ScheduledExecutionApp {

override val firstTime: RichDate = RichDate("2020-04-06")

override val batchIncrement: Duration = Days(1)

override def runOnDateRange(

args: Args

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueID: UniqueID

): Execution[Unit] = {

val maxTopicsPerCountry = args.int("maxTopics", 2000)

val geoPopularTopicsPath: String =

"/user/cassowary/manhattan\_sequence\_files/geo\_popular\_top\_tweet\_impressed\_topics"

// Read engagement logs from the past 7 days

val perTopicEngagementLogData = DAL

.read(PerTopicAggregateEngagementScalaDataset, dateRange.prepend(Days(7)))

.withRemoteReadPolicy(ExplicitLocation(Proc2Atla))

.toTypedPipe

val topicsWithScores =

GeoPopularTopicsApp.getPopularTopicsFromLogs(perTopicEngagementLogData, maxTopicsPerCountry)

val topicsWithEntityScores = topicsWithScores

.mapValues(\_.map {

case (topicid, topicScore) =>

SemanticEntityScore(SemanticCoreEntity(entityId = topicid), topicScore)

})

.mapValues(SemanticCoreEntityScoreList(\_))

val writeKeyValResultExec = topicsWithEntityScores

.map { case (country, topics) => KeyVal(country, topics) }

.writeDALVersionedKeyValExecution(

GeopopularTopTweetImpressedTopicsScalaDataset,

D.Suffix(geoPopularTopicsPath)

)

writeKeyValResultExec

}

}

object GeoPopularTopicsApp {

def getPopularTopicsFromLogs(

engagementLogs: TypedPipe[PerTopicAggregateEngagementMetric],

maxTopics: Int

)(

implicit uniqueId: UniqueID

): TypedPipe[(String, Seq[(SemanticCoreEntityId, Double)])] = {

val numTopicEngagementsRead = Stat("num\_topic\_engagements\_read")

val intermediate = engagementLogs

.map {

case PerTopicAggregateEngagementMetric(

topicId,

dateId,

country,

page,

item,

engagementType,

engagementCount,

algorithmType,

annotationType) =>

numTopicEngagementsRead.inc()

(

topicId,

dateId,

country,

page,

item,

engagementType,

engagementCount,

algorithmType,

annotationType)

}

// We want to find the topics with the most impressed tweets in each country

// This will ensure that the topics suggested as recommendations also have tweets that can be recommended

intermediate

.collect {

case (topicId, \_, Some(country), \_, item, engagementType, engagementCount, \_, \_)

if item == "Tweet" && engagementType == "impression" =>

((country, topicId), engagementCount)

}

.sumByKey // returns country-wise engagements for topics

.map {

case ((country, topicId), totalEngagementCountryCount) =>

(country, (topicId, totalEngagementCountryCount.toDouble))

}

.group

.sortedReverseTake(maxTopics)(Ordering.by(\_.\_2))

.toTypedPipe

}

}