package com.twitter.timelines.prediction.common.aggregates

import com.twitter.dal.client.dataset.KeyValDALDataset

import com.twitter.ml.api.DataRecord

import com.twitter.ml.api.FeatureContext

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

import com.twitter.summingbird.batch.BatchID

import com.twitter.timelines.data\_processing.ml\_util.aggregation\_framework.conversion.CombineCountsPolicy

import com.twitter.timelines.data\_processing.ml\_util.aggregation\_framework.AggregateStore

import com.twitter.timelines.data\_processing.ml\_util.aggregation\_framework.AggregationKey

import com.twitter.timelines.data\_processing.ml\_util.aggregation\_framework.OfflineAggregateDataRecordStore

import scala.collection.JavaConverters.\_

object TimelinesAggregationConfig extends TimelinesAggregationConfigTrait {

override def outputHdfsPath: String = "/user/timelines/processed/aggregates\_v2"

def storeToDatasetMap: Map[String, KeyValDALDataset[

keyval.KeyVal[AggregationKey, (BatchID, DataRecord)]

]] = Map(

AuthorTopicAggregateStore -> AuthorTopicAggregatesScalaDataset,

UserTopicAggregateStore -> UserTopicAggregatesScalaDataset,

UserInferredTopicAggregateStore -> UserInferredTopicAggregatesScalaDataset,

UserAggregateStore -> UserAggregatesScalaDataset,

UserAuthorAggregateStore -> UserAuthorAggregatesScalaDataset,

UserOriginalAuthorAggregateStore -> UserOriginalAuthorAggregatesScalaDataset,

OriginalAuthorAggregateStore -> OriginalAuthorAggregatesScalaDataset,

UserEngagerAggregateStore -> UserEngagerAggregatesScalaDataset,

UserMentionAggregateStore -> UserMentionAggregatesScalaDataset,

TwitterWideUserAggregateStore -> TwitterWideUserAggregatesScalaDataset,

TwitterWideUserAuthorAggregateStore -> TwitterWideUserAuthorAggregatesScalaDataset,

UserRequestHourAggregateStore -> UserRequestHourAggregatesScalaDataset,

UserRequestDowAggregateStore -> UserRequestDowAggregatesScalaDataset,

UserListAggregateStore -> UserListAggregatesScalaDataset,

UserMediaUnderstandingAnnotationAggregateStore -> UserMediaUnderstandingAnnotationAggregatesScalaDataset,

)

override def mkPhysicalStore(store: AggregateStore): AggregateStore = store match {

case s: OfflineAggregateDataRecordStore =>

s.toOfflineAggregateDataRecordStoreWithDAL(storeToDatasetMap(s.name))

case \_ => throw new IllegalArgumentException("Unsupported logical dataset type.")

}

object CombineCountPolicies {

val EngagerCountsPolicy: CombineCountsPolicy = mkCountsPolicy("user\_engager\_aggregate")

val EngagerGoodClickCountsPolicy: CombineCountsPolicy = mkCountsPolicy(

"user\_engager\_good\_click\_aggregate")

val RectweetEngagerCountsPolicy: CombineCountsPolicy =

mkCountsPolicy("rectweet\_user\_engager\_aggregate")

val MentionCountsPolicy: CombineCountsPolicy = mkCountsPolicy("user\_mention\_aggregate")

val RectweetSimclustersTweetCountsPolicy: CombineCountsPolicy =

mkCountsPolicy("rectweet\_user\_simcluster\_tweet\_aggregate")

val UserInferredTopicCountsPolicy: CombineCountsPolicy =

mkCountsPolicy("user\_inferred\_topic\_aggregate")

val UserInferredTopicV2CountsPolicy: CombineCountsPolicy =

mkCountsPolicy("user\_inferred\_topic\_aggregate\_v2")

val UserMediaUnderstandingAnnotationCountsPolicy: CombineCountsPolicy =

mkCountsPolicy("user\_media\_annotation\_aggregate")

private[this] def mkCountsPolicy(prefix: String): CombineCountsPolicy = {

val features = TimelinesAggregationConfig.aggregatesToCompute

.filter(\_.aggregatePrefix == prefix)

.flatMap(\_.allOutputFeatures)

CombineCountsPolicy(

topK = 2,

aggregateContextToPrecompute = new FeatureContext(features.asJava),

hardLimit = Some(20)

)

}

}

}

object TimelinesAggregationCanaryConfig extends TimelinesAggregationConfigTrait {

override def outputHdfsPath: String = "/user/timelines/canaries/processed/aggregates\_v2"

override def mkPhysicalStore(store: AggregateStore): AggregateStore = store match {

case s: OfflineAggregateDataRecordStore =>

s.toOfflineAggregateDataRecordStoreWithDAL(dalDataset = AggregatesCanaryScalaDataset)

case \_ => throw new IllegalArgumentException("Unsupported logical dataset type.")

}

}