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

import com.twitter.ml.api.DataRecord

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

import com.twitter.summingbird.batch.BatchID

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

AggregateStore,

AggregationKey,

OfflineAggregateInjections,

TypedAggregateGroup

}

object TimelinesAggregationKeyValInjections extends TimelinesAggregationConfigTrait {

import OfflineAggregateInjections.getInjection

type KVInjection = KeyValInjection[AggregationKey, (BatchID, DataRecord)]

val AuthorTopic: KVInjection = getInjection(filter(AuthorTopicAggregateStore))

val UserTopic: KVInjection = getInjection(filter(UserTopicAggregateStore))

val UserInferredTopic: KVInjection = getInjection(filter(UserInferredTopicAggregateStore))

val User: KVInjection = getInjection(filter(UserAggregateStore))

val UserAuthor: KVInjection = getInjection(filter(UserAuthorAggregateStore))

val UserOriginalAuthor: KVInjection = getInjection(filter(UserOriginalAuthorAggregateStore))

val OriginalAuthor: KVInjection = getInjection(filter(OriginalAuthorAggregateStore))

val UserEngager: KVInjection = getInjection(filter(UserEngagerAggregateStore))

val UserMention: KVInjection = getInjection(filter(UserMentionAggregateStore))

val TwitterWideUser: KVInjection = getInjection(filter(TwitterWideUserAggregateStore))

val TwitterWideUserAuthor: KVInjection = getInjection(filter(TwitterWideUserAuthorAggregateStore))

val UserRequestHour: KVInjection = getInjection(filter(UserRequestHourAggregateStore))

val UserRequestDow: KVInjection = getInjection(filter(UserRequestDowAggregateStore))

val UserList: KVInjection = getInjection(filter(UserListAggregateStore))

val UserMediaUnderstandingAnnotation: KVInjection = getInjection(

filter(UserMediaUnderstandingAnnotationAggregateStore))

private def filter(storeName: String): Set[TypedAggregateGroup[\_]] = {

val groups = aggregatesToCompute.filter(\_.outputStore.name == storeName)

require(groups.nonEmpty)

groups

}

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

// Since this object is not used to execute any online or offline aggregates job, but is meant

// to store all PDT enabled KeyValInjections, we do not need to construct a physical store.

// We use the identity operation as a default.

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

}