package com.twitter.home\_mixer.product.scored\_tweets.feature\_hydrator.real\_time\_aggregates

import com.google.inject.name.Named

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.home\_mixer.model.HomeFeatures.TopicIdSocialContextFeature

import com.twitter.home\_mixer.param.HomeMixerInjectionNames.TopicCountryEngagementCache

import com.twitter.ml.api.DataRecord

import com.twitter.product\_mixer.component\_library.model.candidate.TweetCandidate

import com.twitter.product\_mixer.core.feature.FeatureWithDefaultOnFailure

import com.twitter.product\_mixer.core.feature.datarecord.DataRecordInAFeature

import com.twitter.product\_mixer.core.model.common.CandidateWithFeatures

import com.twitter.product\_mixer.core.model.common.identifier.FeatureHydratorIdentifier

import com.twitter.product\_mixer.core.pipeline.PipelineQuery

import com.twitter.servo.cache.ReadCache

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

import com.twitter.timelines.prediction.common.aggregates.real\_time.TimelinesOnlineAggregationFeaturesOnlyConfig.\_

import javax.inject.Inject

import javax.inject.Singleton

object TopicCountryEngagementRealTimeAggregateFeature

extends DataRecordInAFeature[TweetCandidate]

with FeatureWithDefaultOnFailure[TweetCandidate, DataRecord] {

override def defaultValue: DataRecord = new DataRecord()

}

@Singleton

class TopicCountryEngagementRealTimeAggregateFeatureHydrator @Inject() (

@Named(TopicCountryEngagementCache) override val client: ReadCache[(Long, String), DataRecord],

override val statsReceiver: StatsReceiver)

extends BaseRealTimeAggregateBulkCandidateFeatureHydrator[(Long, String)] {

override val identifier: FeatureHydratorIdentifier =

FeatureHydratorIdentifier("TopicCountryEngagementRealTimeAggregate")

override val outputFeature: DataRecordInAFeature[TweetCandidate] =

TopicCountryEngagementRealTimeAggregateFeature

override val aggregateGroups: Seq[AggregateGroup] = Seq(

topicCountryRealTimeAggregates

)

override val aggregateGroupToPrefix: Map[AggregateGroup, String] = Map(

topicCountryRealTimeAggregates -> "topic-country\_code.timelines.topic\_country\_engagement\_real\_time\_aggregates."

)

override def keysFromQueryAndCandidates(

query: PipelineQuery,

candidates: Seq[CandidateWithFeatures[TweetCandidate]]

): Seq[Option[(Long, String)]] = {

candidates.map { candidate =>

val maybeTopicId = candidate.features

.getTry(TopicIdSocialContextFeature)

.toOption

.flatten

val maybeCountryCode = query.clientContext.countryCode

for {

topicId <- maybeTopicId

countryCode <- maybeCountryCode

} yield (topicId, countryCode)

}

}

}