package com.twitter.timelines.prediction.common.aggregates.real\_time

import com.twitter.conversions.DurationOps.\_

import com.twitter.ml.api.Feature

import com.twitter.ml.api.constant.SharedFeatures

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

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

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

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

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

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

import com.twitter.timelines.data\_processing.ml\_util.transforms.BinaryUnion

import com.twitter.timelines.data\_processing.ml\_util.transforms.DownsampleTransform

import com.twitter.timelines.data\_processing.ml\_util.transforms.IsNewUserTransform

import com.twitter.timelines.data\_processing.ml\_util.transforms.IsPositionTransform

import com.twitter.timelines.data\_processing.ml\_util.transforms.LogTransform

import com.twitter.timelines.data\_processing.ml\_util.transforms.PositionCase

import com.twitter.timelines.data\_processing.ml\_util.transforms.RichITransform

import com.twitter.timelines.data\_processing.ml\_util.transforms.RichRemoveUnverifiedUserTransform

import com.twitter.timelines.prediction.features.client\_log\_event.ClientLogEventDataRecordFeatures

import com.twitter.timelines.prediction.features.common.CombinedFeatures

import com.twitter.timelines.prediction.features.common.CombinedFeatures.\_

import com.twitter.timelines.prediction.features.common.ProfileLabelFeatures

import com.twitter.timelines.prediction.features.common.SearchLabelFeatures

import com.twitter.timelines.prediction.features.common.TimelinesSharedFeatures

import com.twitter.timelines.prediction.features.common.TimelinesSharedFeatures.IS\_TOP\_FIVE

import com.twitter.timelines.prediction.features.common.TimelinesSharedFeatures.IS\_TOP\_ONE

import com.twitter.timelines.prediction.features.common.TimelinesSharedFeatures.IS\_TOP\_TEN

import com.twitter.timelines.prediction.features.common.TimelinesSharedFeatures.LOG\_POSITION

import com.twitter.timelines.prediction.features.list\_features.ListFeatures

import com.twitter.timelines.prediction.features.recap.RecapFeatures

import com.twitter.util.Duration

import java.lang.{Boolean => JBoolean}

import java.lang.{Long => JLong}

import scala.io.Source

object TimelinesOnlineAggregationUtils {

val TweetLabels: Set[Feature[JBoolean]] = CombinedFeatures.EngagementsRealTime

val TweetCoreLabels: Set[Feature[JBoolean]] = CombinedFeatures.CoreEngagements

val TweetDwellLabels: Set[Feature[JBoolean]] = CombinedFeatures.DwellEngagements

val TweetCoreAndDwellLabels: Set[Feature[JBoolean]] = TweetCoreLabels ++ TweetDwellLabels

val PrivateEngagementLabelsV2: Set[Feature[JBoolean]] = CombinedFeatures.PrivateEngagementsV2

val ProfileCoreLabels: Set[Feature[JBoolean]] = ProfileLabelFeatures.CoreEngagements

val ProfileNegativeEngagementLabels: Set[Feature[JBoolean]] =

ProfileLabelFeatures.NegativeEngagements

val ProfileNegativeEngagementUnionLabels: Set[Feature[JBoolean]] = Set(

ProfileLabelFeatures.IS\_NEGATIVE\_FEEDBACK\_UNION)

val SearchCoreLabels: Set[Feature[JBoolean]] = SearchLabelFeatures.CoreEngagements

val TweetNegativeEngagementLabels: Set[Feature[JBoolean]] =

CombinedFeatures.NegativeEngagementsRealTime

val TweetNegativeEngagementDontLikeLabels: Set[Feature[JBoolean]] =

CombinedFeatures.NegativeEngagementsRealTimeDontLike

val TweetNegativeEngagementSecondaryLabels: Set[Feature[JBoolean]] =

CombinedFeatures.NegativeEngagementsSecondary

val AllTweetNegativeEngagementLabels: Set[Feature[JBoolean]] =

TweetNegativeEngagementLabels ++ TweetNegativeEngagementDontLikeLabels ++ TweetNegativeEngagementSecondaryLabels

val UserAuthorEngagementLabels: Set[Feature[JBoolean]] = CombinedFeatures.UserAuthorEngagements

val ShareEngagementLabels: Set[Feature[JBoolean]] = CombinedFeatures.ShareEngagements

val BookmarkEngagementLabels: Set[Feature[JBoolean]] = CombinedFeatures.BookmarkEngagements

val AllBCEDwellLabels: Set[Feature[JBoolean]] =

CombinedFeatures.TweetDetailDwellEngagements ++ CombinedFeatures.ProfileDwellEngagements ++ CombinedFeatures.FullscreenVideoDwellEngagements

val AllTweetUnionLabels: Set[Feature[JBoolean]] = Set(

CombinedFeatures.IS\_IMPLICIT\_POSITIVE\_FEEDBACK\_UNION,

CombinedFeatures.IS\_EXPLICIT\_POSITIVE\_FEEDBACK\_UNION,

CombinedFeatures.IS\_ALL\_NEGATIVE\_FEEDBACK\_UNION

)

val AllTweetLabels: Set[Feature[JBoolean]] =

TweetLabels ++ TweetCoreAndDwellLabels ++ AllTweetNegativeEngagementLabels ++ ProfileCoreLabels ++ ProfileNegativeEngagementLabels ++ ProfileNegativeEngagementUnionLabels ++ UserAuthorEngagementLabels ++ SearchCoreLabels ++ ShareEngagementLabels ++ BookmarkEngagementLabels ++ PrivateEngagementLabelsV2 ++ AllBCEDwellLabels ++ AllTweetUnionLabels

def addFeatureFilterFromResource(

prodGroup: AggregateGroup,

aggRemovalPath: String

): AggregateGroup = {

val resource = Some(Source.fromResource(aggRemovalPath))

val lines = resource.map(\_.getLines.toSeq)

lines match {

case Some(value) => prodGroup.copy(aggExclusionRegex = value)

case \_ => prodGroup

}

}

}

trait TimelinesOnlineAggregationDefinitionsTrait extends OnlineAggregationConfigTrait {

import TimelinesOnlineAggregationUtils.\_

def inputSource: AggregateSource

def ProductionStore: AggregateStore

def StagingStore: AggregateStore

val TweetFeatures: Set[Feature[\_]] = Set(

ClientLogEventDataRecordFeatures.HasConsumerVideo,

ClientLogEventDataRecordFeatures.PhotoCount

)

val CandidateTweetSourceFeatures: Set[Feature[\_]] = Set(

ClientLogEventDataRecordFeatures.FromRecap,

ClientLogEventDataRecordFeatures.FromRecycled,

ClientLogEventDataRecordFeatures.FromActivity,

ClientLogEventDataRecordFeatures.FromSimcluster,

ClientLogEventDataRecordFeatures.FromErg,

ClientLogEventDataRecordFeatures.FromCroon,

ClientLogEventDataRecordFeatures.FromList,

ClientLogEventDataRecordFeatures.FromRecTopic

)

def createStagingGroup(prodGroup: AggregateGroup): AggregateGroup =

prodGroup.copy(

outputStore = StagingStore

)

// Aggregate user engagements/features by tweet Id.

val tweetEngagement30MinuteCountsProd =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_tweet\_aggregates\_v1",

keys = Set(TimelinesSharedFeatures.SOURCE\_TWEET\_ID),

features = Set.empty,

labels = TweetLabels ++ TweetNegativeEngagementDontLikeLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes),

outputStore = ProductionStore,

includeAnyLabel = false,

includeTimestampFeature = false,

)

// Aggregate user engagements/features by tweet Id.

val tweetVerifiedDontLikeEngagementRealTimeAggregatesProd =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_tweet\_aggregates\_v6",

preTransforms = Seq(RichRemoveUnverifiedUserTransform),

keys = Set(TimelinesSharedFeatures.SOURCE\_TWEET\_ID),

features = Set.empty,

labels = TweetNegativeEngagementDontLikeLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, Duration.Top),

outputStore = ProductionStore,

includeAnyLabel = false,

includeTimestampFeature = false,

)

val tweetNegativeEngagement6HourCounts =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_tweet\_aggregates\_v2",

keys = Set(TimelinesSharedFeatures.SOURCE\_TWEET\_ID),

features = Set.empty,

labels = TweetNegativeEngagementLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes),

outputStore = ProductionStore,

includeAnyLabel = false,

includeTimestampFeature = false,

)

val tweetVerifiedNegativeEngagementCounts =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_tweet\_aggregates\_v7",

preTransforms = Seq(RichRemoveUnverifiedUserTransform),

keys = Set(TimelinesSharedFeatures.SOURCE\_TWEET\_ID),

features = Set.empty,

labels = TweetNegativeEngagementLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, Duration.Top),

outputStore = ProductionStore,

includeAnyLabel = false,

includeTimestampFeature = false,

)

val promotedTweetEngagementRealTimeCounts =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_tweet\_aggregates\_v3.is\_promoted",

preTransforms = Seq(

DownsampleTransform(

negativeSamplingRate = 0.0,

keepLabels = Set(ClientLogEventDataRecordFeatures.IsPromoted))),

keys = Set(TimelinesSharedFeatures.SOURCE\_TWEET\_ID),

features = Set.empty,

labels = TweetCoreAndDwellLabels,

metrics = Set(CountMetric),

halfLives = Set(2.hours, 24.hours),

outputStore = ProductionStore,

includeAnyFeature = false,

includeAnyLabel = false,

includeTimestampFeature = false,

)

/\*\*

\* Aggregate total engagement counts by tweet Id for non-public

\* engagements. Similar to EB's public engagement counts.

\*/

val tweetEngagementTotalCountsProd =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_tweet\_aggregates\_v1",

keys = Set(TimelinesSharedFeatures.SOURCE\_TWEET\_ID),

features = Set.empty,

labels = TweetLabels ++ TweetNegativeEngagementDontLikeLabels,

metrics = Set(CountMetric),

halfLives = Set(Duration.Top),

outputStore = ProductionStore,

includeAnyLabel = false,

includeTimestampFeature = false,

)

val tweetNegativeEngagementTotalCounts =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_tweet\_aggregates\_v2",

keys = Set(TimelinesSharedFeatures.SOURCE\_TWEET\_ID),

features = Set.empty,

labels = TweetNegativeEngagementLabels,

metrics = Set(CountMetric),

halfLives = Set(Duration.Top),

outputStore = ProductionStore,

includeAnyLabel = false,

includeTimestampFeature = false,

)

/\*\*

\* Aggregate tweet features grouped by viewer's user id.

\*/

val userEngagementRealTimeAggregatesProd =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_user\_aggregates\_v1",

keys = Set(SharedFeatures.USER\_ID),

features = TweetFeatures,

labels = TweetLabels ++ TweetNegativeEngagementDontLikeLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes),

outputStore = ProductionStore,

includeAnyLabel = false,

includeTimestampFeature = false,

)

/\*\*

\* Aggregate tweet features grouped by viewer's user id.

\*/

val userEngagementRealTimeAggregatesV2 =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_user\_aggregates\_v2",

keys = Set(SharedFeatures.USER\_ID),

features = ClientLogEventDataRecordFeatures.TweetFeaturesV2,

labels = TweetCoreAndDwellLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, Duration.Top),

outputStore = ProductionStore,

includeAnyFeature = false,

includeAnyLabel = false,

includeTimestampFeature = false,

)

/\*\*

\* Aggregate author's user state features grouped by viewer's user id.

\*/

val userEngagementAuthorUserStateRealTimeAggregates =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_user\_aggregates\_v3",

preTransforms = Seq.empty,

keys = Set(SharedFeatures.USER\_ID),

features = AuthorFeaturesAdapter.UserStateBooleanFeatures,

labels = TweetCoreAndDwellLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, Duration.Top),

outputStore = ProductionStore,

includeAnyFeature = false,

includeAnyLabel = false,

includeTimestampFeature = false,

)

/\*\*

\* Aggregate author's user state features grouped by viewer's user id.

\*/

val userNegativeEngagementAuthorUserStateRealTimeAggregates =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_user\_aggregates\_v4",

preTransforms = Seq.empty,

keys = Set(SharedFeatures.USER\_ID),

features = AuthorFeaturesAdapter.UserStateBooleanFeatures,

labels = TweetNegativeEngagementLabels ++ TweetNegativeEngagementDontLikeLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, Duration.Top),

outputStore = ProductionStore,

includeAnyFeature = false,

includeAnyLabel = false,

includeTimestampFeature = false,

)

/\*\*

\* Aggregate tweet features grouped by viewer's user id, with 48 hour halfLife.

\*/

val userEngagement48HourRealTimeAggregatesProd =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_user\_aggregates\_v5",

keys = Set(SharedFeatures.USER\_ID),

features = TweetFeatures,

labels = TweetLabels ++ TweetNegativeEngagementDontLikeLabels,

metrics = Set(CountMetric),

halfLives = Set(48.hours),

outputStore = ProductionStore,

includeAnyLabel = false,

includeTimestampFeature = false,

)

/\*\*

\* Aggregate author's user state features grouped by viewer's user id.

\*/

val userNegativeEngagementAuthorUserState72HourRealTimeAggregates =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_user\_aggregates\_v6",

preTransforms = Seq.empty,

keys = Set(SharedFeatures.USER\_ID),

features = AuthorFeaturesAdapter.UserStateBooleanFeatures,

labels = TweetNegativeEngagementLabels ++ TweetNegativeEngagementDontLikeLabels,

metrics = Set(CountMetric),

halfLives = Set(72.hours),

outputStore = ProductionStore,

includeAnyFeature = false,

includeAnyLabel = false,

includeTimestampFeature = false,

)

/\*\*

\* Aggregate features grouped by source author id: for each author, aggregate features are created

\* to quantify engagements (fav, reply, etc.) which tweets of the author has received.

\*/

val authorEngagementRealTimeAggregatesProd =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_author\_aggregates\_v1",

keys = Set(TimelinesSharedFeatures.SOURCE\_AUTHOR\_ID),

features = Set.empty,

labels = TweetLabels ++ TweetNegativeEngagementDontLikeLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, Duration.Top),

outputStore = ProductionStore,

includeAnyLabel = false,

includeTimestampFeature = false,

)

/\*\*

\* Aggregate features grouped by source author id: for each author, aggregate features are created

\* to quantify negative engagements (mute, block, etc.) which tweets of the author has received.

\*

\* This aggregate group is not used in Home, but it is used in Follow Recommendation Service so need to keep it for now.

\*

\*/

val authorNegativeEngagementRealTimeAggregatesProd =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_author\_aggregates\_v2",

keys = Set(TimelinesSharedFeatures.SOURCE\_AUTHOR\_ID),

features = Set.empty,

labels = TweetNegativeEngagementLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, Duration.Top),

outputStore = ProductionStore,

includeAnyLabel = false,

includeTimestampFeature = false,

)

/\*\*

\* Aggregate features grouped by source author id: for each author, aggregate features are created

\* to quantify negative engagements (don't like) which tweets of the author has received from

\* verified users.

\*/

val authorVerifiedNegativeEngagementRealTimeAggregatesProd =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_author\_aggregates\_v3",

preTransforms = Seq(RichRemoveUnverifiedUserTransform),

keys = Set(TimelinesSharedFeatures.SOURCE\_AUTHOR\_ID),

features = Set.empty,

labels = TweetNegativeEngagementDontLikeLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes),

outputStore = ProductionStore,

includeAnyLabel = false,

includeTimestampFeature = false,

)

/\*\*

\* Aggregate tweet features grouped by topic id.

\*/

val topicEngagementRealTimeAggregatesProd =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_topic\_aggregates\_v1",

keys = Set(TimelinesSharedFeatures.TOPIC\_ID),

features = Set.empty,

labels = TweetLabels ++ AllTweetNegativeEngagementLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, Duration.Top),

outputStore = ProductionStore,

includeAnyLabel = false,

includeTimestampFeature = false,

)

/\*\*

\* Aggregate user engagements / user state by topic id.

\*/

val topicEngagementUserStateRealTimeAggregatesProd =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_topic\_aggregates\_v2",

keys = Set(TimelinesSharedFeatures.TOPIC\_ID),

features = UserFeaturesAdapter.UserStateBooleanFeatures,

labels = TweetCoreAndDwellLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, Duration.Top),

outputStore = ProductionStore,

includeAnyFeature = false,

includeAnyLabel = false,

includeTimestampFeature = false,

)

/\*\*

\* Aggregate user negative engagements / user state by topic id.

\*/

val topicNegativeEngagementUserStateRealTimeAggregatesProd =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_topic\_aggregates\_v3",

keys = Set(TimelinesSharedFeatures.TOPIC\_ID),

features = UserFeaturesAdapter.UserStateBooleanFeatures,

labels = TweetNegativeEngagementLabels ++ TweetNegativeEngagementDontLikeLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, Duration.Top),

outputStore = ProductionStore,

includeAnyFeature = false,

includeAnyLabel = false,

includeTimestampFeature = false,

)

/\*\*

\* Aggregate tweet features grouped by topic id like real\_time\_topic\_aggregates\_v1 but 24hour halfLife

\*/

val topicEngagement24HourRealTimeAggregatesProd =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_topic\_aggregates\_v4",

keys = Set(TimelinesSharedFeatures.TOPIC\_ID),

features = Set.empty,

labels = TweetLabels ++ AllTweetNegativeEngagementLabels,

metrics = Set(CountMetric),

halfLives = Set(24.hours),

outputStore = ProductionStore,

includeAnyLabel = false,

includeTimestampFeature = false,

)

// Aggregate user engagements / user state by tweet Id.

val tweetEngagementUserStateRealTimeAggregatesProd =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_tweet\_aggregates\_v3",

keys = Set(TimelinesSharedFeatures.SOURCE\_TWEET\_ID),

features = UserFeaturesAdapter.UserStateBooleanFeatures,

labels = TweetCoreAndDwellLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, Duration.Top),

outputStore = ProductionStore,

includeAnyFeature = false,

includeAnyLabel = false,

includeTimestampFeature = false,

)

// Aggregate user engagements / user gender by tweet Id.

val tweetEngagementGenderRealTimeAggregatesProd =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_tweet\_aggregates\_v4",

keys = Set(TimelinesSharedFeatures.SOURCE\_TWEET\_ID),

features = UserFeaturesAdapter.GenderBooleanFeatures,

labels =

TweetCoreAndDwellLabels ++ TweetNegativeEngagementLabels ++ TweetNegativeEngagementDontLikeLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, Duration.Top),

outputStore = ProductionStore,

includeAnyFeature = false,

includeAnyLabel = false,

includeTimestampFeature = false,

)

// Aggregate user negative engagements / user state by tweet Id.

val tweetNegativeEngagementUserStateRealTimeAggregates =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_tweet\_aggregates\_v5",

keys = Set(TimelinesSharedFeatures.SOURCE\_TWEET\_ID),

features = UserFeaturesAdapter.UserStateBooleanFeatures,

labels = TweetNegativeEngagementLabels ++ TweetNegativeEngagementDontLikeLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, Duration.Top),

outputStore = ProductionStore,

includeAnyFeature = false,

includeAnyLabel = false,

includeTimestampFeature = false,

)

// Aggregate user negative engagements / user state by tweet Id.

val tweetVerifiedNegativeEngagementUserStateRealTimeAggregates =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_tweet\_aggregates\_v8",

preTransforms = Seq(RichRemoveUnverifiedUserTransform),

keys = Set(TimelinesSharedFeatures.SOURCE\_TWEET\_ID),

features = UserFeaturesAdapter.UserStateBooleanFeatures,

labels = TweetNegativeEngagementLabels ++ TweetNegativeEngagementDontLikeLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, Duration.Top),

outputStore = ProductionStore,

includeAnyFeature = false,

includeAnyLabel = false,

includeTimestampFeature = false,

)

/\*\*

\* Aggregate tweet engagement labels and candidate tweet source features grouped by user id.

\*/

val userCandidateTweetSourceEngagementRealTimeAggregatesProd =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_user\_candidate\_tweet\_source\_aggregates\_v1",

keys = Set(SharedFeatures.USER\_ID),

features = CandidateTweetSourceFeatures,

labels = TweetCoreAndDwellLabels ++ NegativeEngagementsRealTimeDontLike,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, Duration.Top),

outputStore = ProductionStore,

includeAnyFeature = false,

includeAnyLabel = false,

includeTimestampFeature = false,

)

/\*\*

\* Aggregate tweet engagement labels and candidate tweet source features grouped by user id.

\*/

val userCandidateTweetSourceEngagement48HourRealTimeAggregatesProd =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_user\_candidate\_tweet\_source\_aggregates\_v2",

keys = Set(SharedFeatures.USER\_ID),

features = CandidateTweetSourceFeatures,

labels = TweetCoreAndDwellLabels ++ NegativeEngagementsRealTimeDontLike,

metrics = Set(CountMetric),

halfLives = Set(48.hours),

outputStore = ProductionStore,

includeAnyFeature = false,

includeAnyLabel = false,

includeTimestampFeature = false,

)

/\*\*

\* Aggregate tweet features grouped by viewer's user id on Profile engagements

\*/

val userProfileEngagementRealTimeAggregates =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "profile\_real\_time\_user\_aggregates\_v1",

preTransforms = Seq(IsNewUserTransform),

keys = Set(SharedFeatures.USER\_ID),

features = TweetFeatures,

labels = ProfileCoreLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, Duration.Top),

outputStore = ProductionStore,

includeAnyFeature = true,

includeAnyLabel = false,

includeTimestampFeature = false,

)

val NegativeEngagementsUnionTransform = RichITransform(

BinaryUnion(

featuresToUnify = ProfileNegativeEngagementLabels,

outputFeature = ProfileLabelFeatures.IS\_NEGATIVE\_FEEDBACK\_UNION

))

/\*\*

\* Aggregate tweet features grouped by viewer's user id on Profile negative engagements.

\*/

val userProfileNegativeEngagementRealTimeAggregates =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "profile\_negative\_engagement\_real\_time\_user\_aggregates\_v1",

preTransforms = Seq(NegativeEngagementsUnionTransform),

keys = Set(SharedFeatures.USER\_ID),

features = Set.empty,

labels = ProfileNegativeEngagementLabels ++ ProfileNegativeEngagementUnionLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, 72.hours, 14.day),

outputStore = ProductionStore,

includeAnyFeature = true,

includeAnyLabel = false,

includeTimestampFeature = false,

)

/\*\*

\* Aggregate tweet features grouped by viewer's and author's user ids and on Profile engagements

\*/

val userAuthorProfileEngagementRealTimeAggregates =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "user\_author\_profile\_real\_time\_aggregates\_v1",

keys = Set(SharedFeatures.USER\_ID, TimelinesSharedFeatures.SOURCE\_AUTHOR\_ID),

features = Set.empty,

labels = ProfileCoreLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, 24.hours, 72.hours),

outputStore = ProductionStore,

includeAnyFeature = true,

includeAnyLabel = false,

includeTimestampFeature = false,

)

/\*\*

\* Aggregate tweet features grouped by viewer's and author's user ids and on negative Profile engagements

\*/

val userAuthorProfileNegativeEngagementRealTimeAggregates =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "user\_author\_profile\_negative\_engagement\_real\_time\_aggregates\_v1",

preTransforms = Seq(NegativeEngagementsUnionTransform),

keys = Set(SharedFeatures.USER\_ID, TimelinesSharedFeatures.SOURCE\_AUTHOR\_ID),

features = Set.empty,

labels = ProfileNegativeEngagementUnionLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, 72.hours, 14.day),

outputStore = ProductionStore,

includeAnyFeature = true,

includeAnyLabel = false,

includeTimestampFeature = false,

)

val newUserAuthorEngagementRealTimeAggregatesProd =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_new\_user\_author\_aggregates\_v1",

preTransforms = Seq(IsNewUserTransform),

keys = Set(SharedFeatures.USER\_ID, TimelinesSharedFeatures.SOURCE\_AUTHOR\_ID),

features = Set.empty,

labels = TweetCoreAndDwellLabels ++ Set(

IS\_CLICKED,

IS\_PROFILE\_CLICKED,

IS\_PHOTO\_EXPANDED

),

metrics = Set(CountMetric),

halfLives = Set(30.minutes, Duration.Top),

outputStore = ProductionStore,

includeAnyFeature = true,

includeAnyLabel = false,

includeTimestampFeature = false,

)

val userAuthorEngagementRealTimeAggregatesProd = {

// Computing user-author real-time aggregates is very expensive so we

// take the union of all major negative feedback engagements to create

// a single negtive label for aggregation. We also include a number of

// core positive engagements.

val BinaryUnionNegativeEngagements =

BinaryUnion(

featuresToUnify = AllTweetNegativeEngagementLabels,

outputFeature = IS\_NEGATIVE\_FEEDBACK\_UNION

)

val BinaryUnionNegativeEngagementsTransform = RichITransform(BinaryUnionNegativeEngagements)

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_user\_author\_aggregates\_v1",

preTransforms = Seq(BinaryUnionNegativeEngagementsTransform),

keys = Set(SharedFeatures.USER\_ID, TimelinesSharedFeatures.SOURCE\_AUTHOR\_ID),

features = Set.empty,

labels = UserAuthorEngagementLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, 1.day),

outputStore = ProductionStore,

includeAnyFeature = true,

includeAnyLabel = false,

includeTimestampFeature = false,

)

}

/\*\*

\* Aggregate tweet features grouped by list id.

\*/

val listEngagementRealTimeAggregatesProd =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_list\_aggregates\_v1",

keys = Set(ListFeatures.LIST\_ID),

features = Set.empty,

labels =

TweetCoreAndDwellLabels ++ TweetNegativeEngagementLabels ++ TweetNegativeEngagementDontLikeLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, Duration.Top),

outputStore = ProductionStore,

includeAnyLabel = false,

includeTimestampFeature = false,

)

// Aggregate features grouped by topic of tweet and country from user's location

val topicCountryRealTimeAggregates =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_topic\_country\_aggregates\_v1",

keys = Set(TimelinesSharedFeatures.TOPIC\_ID, UserFeaturesAdapter.USER\_COUNTRY\_ID),

features = Set.empty,

labels =

TweetCoreAndDwellLabels ++ AllTweetNegativeEngagementLabels ++ PrivateEngagementLabelsV2 ++ ShareEngagementLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, 72.hours),

outputStore = ProductionStore,

includeAnyLabel = false,

includeTimestampFeature = false,

)

// Aggregate features grouped by TweetId\_Country from user's location

val tweetCountryRealTimeAggregates =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_tweet\_country\_aggregates\_v1",

keys = Set(TimelinesSharedFeatures.SOURCE\_TWEET\_ID, UserFeaturesAdapter.USER\_COUNTRY\_ID),

features = Set.empty,

labels = TweetCoreAndDwellLabels ++ AllTweetNegativeEngagementLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, Duration.Top),

outputStore = ProductionStore,

includeAnyLabel = true,

includeTimestampFeature = false,

)

// Additional aggregate features grouped by TweetId\_Country from user's location

val tweetCountryPrivateEngagementsRealTimeAggregates =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_tweet\_country\_aggregates\_v2",

keys = Set(TimelinesSharedFeatures.SOURCE\_TWEET\_ID, UserFeaturesAdapter.USER\_COUNTRY\_ID),

features = Set.empty,

labels = PrivateEngagementLabelsV2 ++ ShareEngagementLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, 72.hours),

outputStore = ProductionStore,

includeAnyLabel = false,

includeTimestampFeature = false,

)

// Aggregate features grouped by TweetId\_Country from user's location

val tweetCountryVerifiedNegativeEngagementsRealTimeAggregates =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_tweet\_country\_aggregates\_v3",

preTransforms = Seq(RichRemoveUnverifiedUserTransform),

keys = Set(TimelinesSharedFeatures.SOURCE\_TWEET\_ID, UserFeaturesAdapter.USER\_COUNTRY\_ID),

features = Set.empty,

labels = AllTweetNegativeEngagementLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, Duration.Top),

outputStore = ProductionStore,

includeAnyLabel = true,

includeTimestampFeature = false,

)

object positionTranforms extends IsPositionTransform {

override val isInPositionRangeFeature: Seq[PositionCase] =

Seq(PositionCase(1, IS\_TOP\_ONE), PositionCase(5, IS\_TOP\_FIVE), PositionCase(10, IS\_TOP\_TEN))

override val decodedPositionFeature: Feature.Discrete =

ClientLogEventDataRecordFeatures.InjectedPosition

}

val userPositionEngagementsCountsProd =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_position\_based\_user\_aggregates\_v1",

keys = Set(SharedFeatures.USER\_ID),

features = Set(IS\_TOP\_ONE, IS\_TOP\_FIVE, IS\_TOP\_TEN),

labels = TweetCoreAndDwellLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, 24.hours),

outputStore = ProductionStore,

preTransforms = Seq(positionTranforms),

includeAnyLabel = false,

includeAnyFeature = false,

includeTimestampFeature = false,

)

val userPositionEngagementsSumProd =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_position\_based\_user\_sum\_aggregates\_v2",

keys = Set(SharedFeatures.USER\_ID),

features = Set(LOG\_POSITION),

labels = TweetCoreAndDwellLabels,

metrics = Set(SumMetric),

halfLives = Set(30.minutes, 24.hours),

outputStore = ProductionStore,

preTransforms =

Seq(new LogTransform(ClientLogEventDataRecordFeatures.InjectedPosition, LOG\_POSITION)),

includeAnyLabel = false,

includeAnyFeature = false,

includeTimestampFeature = false,

)

// Aggregates for share engagements

val tweetShareEngagementsRealTimeAggregates =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_tweet\_share\_aggregates\_v1",

keys = Set(TimelinesSharedFeatures.SOURCE\_TWEET\_ID),

features = Set.empty,

labels = ShareEngagementLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, 24.hours),

outputStore = ProductionStore,

includeAnyLabel = false,

includeTimestampFeature = false,

)

val userShareEngagementsRealTimeAggregates =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_user\_share\_aggregates\_v1",

keys = Set(SharedFeatures.USER\_ID),

features = Set.empty,

labels = ShareEngagementLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, 24.hours),

outputStore = ProductionStore,

includeAnyLabel = false,

includeTimestampFeature = false,

)

val userAuthorShareEngagementsRealTimeAggregates =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_user\_author\_share\_aggregates\_v1",

keys = Set(SharedFeatures.USER\_ID, TimelinesSharedFeatures.SOURCE\_AUTHOR\_ID),

features = Set.empty,

labels = ShareEngagementLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, 24.hours),

outputStore = ProductionStore,

includeAnyFeature = true,

includeAnyLabel = false,

includeTimestampFeature = false,

)

val topicShareEngagementsRealTimeAggregates =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_topic\_share\_aggregates\_v1",

keys = Set(TimelinesSharedFeatures.TOPIC\_ID),

features = Set.empty,

labels = ShareEngagementLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, 24.hours),

outputStore = ProductionStore,

includeAnyLabel = false,

includeTimestampFeature = false,

)

val authorShareEngagementsRealTimeAggregates =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_author\_share\_aggregates\_v1",

keys = Set(TimelinesSharedFeatures.SOURCE\_AUTHOR\_ID),

features = Set.empty,

labels = ShareEngagementLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, 24.hours),

outputStore = ProductionStore,

includeAnyLabel = false,

includeTimestampFeature = false,

)

// Bookmark RTAs

val tweetBookmarkEngagementsRealTimeAggregates =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_tweet\_bookmark\_aggregates\_v1",

keys = Set(TimelinesSharedFeatures.SOURCE\_TWEET\_ID),

features = Set.empty,

labels = BookmarkEngagementLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, 24.hours),

outputStore = ProductionStore,

includeAnyLabel = false,

includeTimestampFeature = false,

)

val userBookmarkEngagementsRealTimeAggregates =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_user\_bookmark\_aggregates\_v1",

keys = Set(SharedFeatures.USER\_ID),

features = Set.empty,

labels = BookmarkEngagementLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, 24.hours),

outputStore = ProductionStore,

includeAnyLabel = false,

includeTimestampFeature = false,

)

val userAuthorBookmarkEngagementsRealTimeAggregates =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_user\_author\_bookmark\_aggregates\_v1",

keys = Set(SharedFeatures.USER\_ID, TimelinesSharedFeatures.SOURCE\_AUTHOR\_ID),

features = Set.empty,

labels = BookmarkEngagementLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, 24.hours),

outputStore = ProductionStore,

includeAnyFeature = true,

includeAnyLabel = false,

includeTimestampFeature = false,

)

val authorBookmarkEngagementsRealTimeAggregates =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_author\_bookmark\_aggregates\_v1",

keys = Set(TimelinesSharedFeatures.SOURCE\_AUTHOR\_ID),

features = Set.empty,

labels = BookmarkEngagementLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, 24.hours),

outputStore = ProductionStore,

includeAnyLabel = false,

includeTimestampFeature = false,

)

/\*\*

\* Aggregate on user level dwell labels from BCE

\*/

val userBCEDwellEngagementsRealTimeAggregates =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_user\_bce\_dwell\_aggregates",

keys = Set(SharedFeatures.USER\_ID),

features = Set.empty,

labels = AllBCEDwellLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, 24.hours),

outputStore = ProductionStore,

includeAnyLabel = false,

includeTimestampFeature = false,

)

/\*\*

\* Aggregate on tweet level dwell labels from BCE

\*/

val tweetBCEDwellEngagementsRealTimeAggregates =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_tweet\_bce\_dwell\_aggregates",

keys = Set(TimelinesSharedFeatures.SOURCE\_TWEET\_ID),

features = Set.empty,

labels = AllBCEDwellLabels,

metrics = Set(CountMetric),

halfLives = Set(30.minutes, 24.hours),

outputStore = ProductionStore,

includeAnyLabel = false,

includeTimestampFeature = false,

)

val ImplicitPositiveEngagementsUnionTransform = RichITransform(

BinaryUnion(

featuresToUnify = CombinedFeatures.ImplicitPositiveEngagements,

outputFeature = CombinedFeatures.IS\_IMPLICIT\_POSITIVE\_FEEDBACK\_UNION

)

)

val ExplicitPositiveEngagementsUnionTransform = RichITransform(

BinaryUnion(

featuresToUnify = CombinedFeatures.ExplicitPositiveEngagements,

outputFeature = CombinedFeatures.IS\_EXPLICIT\_POSITIVE\_FEEDBACK\_UNION

)

)

val AllNegativeEngagementsUnionTransform = RichITransform(

BinaryUnion(

featuresToUnify = CombinedFeatures.AllNegativeEngagements,

outputFeature = CombinedFeatures.IS\_ALL\_NEGATIVE\_FEEDBACK\_UNION

)

)

/\*\*

\* Aggregate features for author content preference

\*/

val authorContentPreferenceRealTimeAggregates =

AggregateGroup(

inputSource = inputSource,

aggregatePrefix = "real\_time\_author\_content\_preference\_aggregates",

preTransforms = Seq(

ImplicitPositiveEngagementsUnionTransform,

ExplicitPositiveEngagementsUnionTransform,

AllNegativeEngagementsUnionTransform),

keys = Set(TimelinesSharedFeatures.SOURCE\_AUTHOR\_ID),

features =

ClientLogEventDataRecordFeatures.AuthorContentPreferenceTweetTypeFeatures ++ AuthorFeaturesAdapter.UserStateBooleanFeatures,

labels = AllTweetUnionLabels,

metrics = Set(CountMetric),

halfLives = Set(24.hours),

outputStore = ProductionStore,

includeAnyLabel = false,

includeAnyFeature = false,

)

val FeaturesGeneratedByPreTransforms = Set(LOG\_POSITION, IS\_TOP\_TEN, IS\_TOP\_FIVE, IS\_TOP\_ONE)

val ProdAggregateGroups = Set(

tweetEngagement30MinuteCountsProd,

tweetEngagementTotalCountsProd,

tweetNegativeEngagement6HourCounts,

tweetNegativeEngagementTotalCounts,

userEngagementRealTimeAggregatesProd,

userEngagement48HourRealTimeAggregatesProd,

userNegativeEngagementAuthorUserStateRealTimeAggregates,

userNegativeEngagementAuthorUserState72HourRealTimeAggregates,

authorEngagementRealTimeAggregatesProd,

topicEngagementRealTimeAggregatesProd,

topicEngagement24HourRealTimeAggregatesProd,

tweetEngagementUserStateRealTimeAggregatesProd,

tweetNegativeEngagementUserStateRealTimeAggregates,

userProfileEngagementRealTimeAggregates,

newUserAuthorEngagementRealTimeAggregatesProd,

userAuthorEngagementRealTimeAggregatesProd,

listEngagementRealTimeAggregatesProd,

tweetCountryRealTimeAggregates,

tweetShareEngagementsRealTimeAggregates,

userShareEngagementsRealTimeAggregates,

userAuthorShareEngagementsRealTimeAggregates,

topicShareEngagementsRealTimeAggregates,

authorShareEngagementsRealTimeAggregates,

tweetBookmarkEngagementsRealTimeAggregates,

userBookmarkEngagementsRealTimeAggregates,

userAuthorBookmarkEngagementsRealTimeAggregates,

authorBookmarkEngagementsRealTimeAggregates,

topicCountryRealTimeAggregates,

tweetCountryPrivateEngagementsRealTimeAggregates,

userBCEDwellEngagementsRealTimeAggregates,

tweetBCEDwellEngagementsRealTimeAggregates,

authorContentPreferenceRealTimeAggregates,

authorVerifiedNegativeEngagementRealTimeAggregatesProd,

tweetVerifiedDontLikeEngagementRealTimeAggregatesProd,

tweetVerifiedNegativeEngagementCounts,

tweetVerifiedNegativeEngagementUserStateRealTimeAggregates,

tweetCountryVerifiedNegativeEngagementsRealTimeAggregates

).map(

addFeatureFilterFromResource(

\_,

"com/twitter/timelines/prediction/common/aggregates/real\_time/aggregates\_to\_drop.txt"))

val StagingAggregateGroups = ProdAggregateGroups.map(createStagingGroup)

/\*\*

\* Contains the fully typed aggregate groups from which important

\* values can be derived e.g. the features to be computed, halflives etc.

\*/

override val ProdAggregates = ProdAggregateGroups.flatMap(\_.buildTypedAggregateGroups())

override val StagingAggregates = StagingAggregateGroups.flatMap(\_.buildTypedAggregateGroups())

override val ProdCommonAggregates = ProdAggregates

.filter(\_.keysToAggregate == Set(SharedFeatures.USER\_ID))

/\*\*

\* This defines the set of selected features from a candidate

\* that we'd like to send to the served features cache by TLM.

\* These should include interesting and necessary features that

\* cannot be extracted from LogEvents only by the real-time aggregates

\* job. If you are adding new AggregateGroups requiring TLM-side

\* candidate features, make sure to add them here.

\*/

val candidateFeaturesToCache: Set[Feature[\_]] = Set(

TimelinesSharedFeatures.SOURCE\_AUTHOR\_ID,

RecapFeatures.HASHTAGS,

RecapFeatures.MENTIONED\_SCREEN\_NAMES,

RecapFeatures.URL\_DOMAINS

)

}

/\*\*

\* This config should only be used to access the aggregate features constructed by the

\* aggregation config, and not for implementing an online real-time aggregates job.

\*/

object TimelinesOnlineAggregationFeaturesOnlyConfig

extends TimelinesOnlineAggregationDefinitionsTrait {

private[real\_time] case class DummyAggregateSource(name: String, timestampFeature: Feature[JLong])

extends AggregateSource

private[real\_time] case class DummyAggregateStore(name: String) extends AggregateStore

override lazy val inputSource = DummyAggregateSource(

name = "timelines\_rta",

timestampFeature = SharedFeatures.TIMESTAMP

)

override lazy val ProductionStore = DummyAggregateStore("timelines\_rta")

override lazy val StagingStore = DummyAggregateStore("timelines\_rta")

override lazy val AggregatesToCompute = ProdAggregates ++ StagingAggregates

}