package com.twitter.cr\_mixer.source\_signal

import com.twitter.cr\_mixer.param.GlobalParams

import com.twitter.cr\_mixer.param.GoodProfileClickParams

import com.twitter.cr\_mixer.param.GoodTweetClickParams

import com.twitter.cr\_mixer.param.RealGraphOonParams

import com.twitter.cr\_mixer.param.RecentFollowsParams

import com.twitter.cr\_mixer.param.RecentNegativeSignalParams

import com.twitter.cr\_mixer.param.RecentNotificationsParams

import com.twitter.cr\_mixer.param.RecentOriginalTweetsParams

import com.twitter.cr\_mixer.param.RecentReplyTweetsParams

import com.twitter.cr\_mixer.param.RecentRetweetsParams

import com.twitter.cr\_mixer.param.RecentTweetFavoritesParams

import com.twitter.cr\_mixer.param.RepeatedProfileVisitsParams

import com.twitter.cr\_mixer.param.TweetSharesParams

import com.twitter.cr\_mixer.param.UnifiedUSSSignalParams

import com.twitter.cr\_mixer.param.VideoViewTweetsParams

import com.twitter.cr\_mixer.source\_signal.UssStore.Query

import com.twitter.cr\_mixer.thriftscala.SourceType

import com.twitter.finagle.stats.StatsReceiver

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

import com.twitter.storehaus.ReadableStore

import com.twitter.usersignalservice.thriftscala.{Signal => UssSignal}

import com.twitter.usersignalservice.thriftscala.SignalType

import javax.inject.Singleton

import com.twitter.timelines.configapi

import com.twitter.timelines.configapi.Params

import com.twitter.usersignalservice.thriftscala.BatchSignalRequest

import com.twitter.usersignalservice.thriftscala.BatchSignalResponse

import com.twitter.usersignalservice.thriftscala.SignalRequest

import com.twitter.util.Future

import com.twitter.cr\_mixer.thriftscala.Product

import com.twitter.usersignalservice.thriftscala.ClientIdentifier

@Singleton

case class UssStore(

stratoStore: ReadableStore[BatchSignalRequest, BatchSignalResponse],

statsReceiver: StatsReceiver)

extends ReadableStore[Query, Seq[(SignalType, Seq[UssSignal])]] {

import com.twitter.cr\_mixer.source\_signal.UssStore.\_

override def get(query: Query): Future[Option[Seq[(SignalType, Seq[UssSignal])]]] = {

val ussClientIdentifier = query.product match {

case Product.Home =>

ClientIdentifier.CrMixerHome

case Product.Notifications =>

ClientIdentifier.CrMixerNotifications

case Product.Email =>

ClientIdentifier.CrMixerEmail

case \_ =>

ClientIdentifier.Unknown

}

val batchSignalRequest =

BatchSignalRequest(

query.userId,

buildUserSignalServiceRequests(query.params),

Some(ussClientIdentifier))

stratoStore

.get(batchSignalRequest)

.map {

\_.map { batchSignalResponse =>

batchSignalResponse.signalResponse.toSeq.map {

case (signalType, ussSignals) =>

(signalType, ussSignals)

}

}

}

}

private def buildUserSignalServiceRequests(

param: Params,

): Seq[SignalRequest] = {

val unifiedMaxSourceKeyNum = param(GlobalParams.UnifiedMaxSourceKeyNum)

val goodTweetClickMaxSignalNum = param(GoodTweetClickParams.MaxSignalNumParam)

val aggrTweetMaxSourceKeyNum = param(UnifiedUSSSignalParams.UnifiedTweetSourceNumberParam)

val aggrProducerMaxSourceKeyNum = param(UnifiedUSSSignalParams.UnifiedProducerSourceNumberParam)

val maybeRecentTweetFavorite =

if (param(RecentTweetFavoritesParams.EnableSourceParam))

Some(SignalRequest(Some(unifiedMaxSourceKeyNum), SignalType.TweetFavorite))

else None

val maybeRecentRetweet =

if (param(RecentRetweetsParams.EnableSourceParam))

Some(SignalRequest(Some(unifiedMaxSourceKeyNum), SignalType.Retweet))

else None

val maybeRecentReply =

if (param(RecentReplyTweetsParams.EnableSourceParam))

Some(SignalRequest(Some(unifiedMaxSourceKeyNum), SignalType.Reply))

else None

val maybeRecentOriginalTweet =

if (param(RecentOriginalTweetsParams.EnableSourceParam))

Some(SignalRequest(Some(unifiedMaxSourceKeyNum), SignalType.OriginalTweet))

else None

val maybeRecentFollow =

if (param(RecentFollowsParams.EnableSourceParam))

Some(SignalRequest(Some(unifiedMaxSourceKeyNum), SignalType.AccountFollow))

else None

val maybeRepeatedProfileVisits =

if (param(RepeatedProfileVisitsParams.EnableSourceParam))

Some(

SignalRequest(

Some(unifiedMaxSourceKeyNum),

param(RepeatedProfileVisitsParams.ProfileMinVisitType).signalType))

else None

val maybeRecentNotifications =

if (param(RecentNotificationsParams.EnableSourceParam))

Some(SignalRequest(Some(unifiedMaxSourceKeyNum), SignalType.NotificationOpenAndClickV1))

else None

val maybeTweetShares =

if (param(TweetSharesParams.EnableSourceParam)) {

Some(SignalRequest(Some(unifiedMaxSourceKeyNum), SignalType.TweetShareV1))

} else None

val maybeRealGraphOon =

if (param(RealGraphOonParams.EnableSourceParam)) {

Some(SignalRequest(Some(unifiedMaxSourceKeyNum), SignalType.RealGraphOon))

} else None

val maybeGoodTweetClick =

if (param(GoodTweetClickParams.EnableSourceParam))

Some(

SignalRequest(

Some(goodTweetClickMaxSignalNum),

param(GoodTweetClickParams.ClickMinDwellTimeType).signalType))

else None

val maybeVideoViewTweets =

if (param(VideoViewTweetsParams.EnableSourceParam)) {

Some(

SignalRequest(

Some(unifiedMaxSourceKeyNum),

param(VideoViewTweetsParams.VideoViewTweetTypeParam).signalType))

} else None

val maybeGoodProfileClick =

if (param(GoodProfileClickParams.EnableSourceParam))

Some(

SignalRequest(

Some(unifiedMaxSourceKeyNum),

param(GoodProfileClickParams.ClickMinDwellTimeType).signalType))

else None

val maybeAggTweetSignal =

if (param(UnifiedUSSSignalParams.EnableTweetAggSourceParam))

Some(

SignalRequest(

Some(aggrTweetMaxSourceKeyNum),

param(UnifiedUSSSignalParams.TweetAggTypeParam).signalType

)

)

else None

val maybeAggProducerSignal =

if (param(UnifiedUSSSignalParams.EnableProducerAggSourceParam))

Some(

SignalRequest(

Some(aggrProducerMaxSourceKeyNum),

param(UnifiedUSSSignalParams.ProducerAggTypeParam).signalType

)

)

else None

// negative signals

val maybeNegativeSignals = if (param(RecentNegativeSignalParams.EnableSourceParam)) {

EnabledNegativeSignalTypes

.map(negativeSignal => SignalRequest(Some(unifiedMaxSourceKeyNum), negativeSignal)).toSeq

} else Seq.empty

val allPositiveSignals =

if (param(UnifiedUSSSignalParams.ReplaceIndividualUSSSourcesParam))

Seq(

maybeRecentOriginalTweet,

maybeRecentNotifications,

maybeRealGraphOon,

maybeGoodTweetClick,

maybeGoodProfileClick,

maybeAggProducerSignal,

maybeAggTweetSignal,

)

else

Seq(

maybeRecentTweetFavorite,

maybeRecentRetweet,

maybeRecentReply,

maybeRecentOriginalTweet,

maybeRecentFollow,

maybeRepeatedProfileVisits,

maybeRecentNotifications,

maybeTweetShares,

maybeRealGraphOon,

maybeGoodTweetClick,

maybeVideoViewTweets,

maybeGoodProfileClick,

maybeAggProducerSignal,

maybeAggTweetSignal,

)

allPositiveSignals.flatten ++ maybeNegativeSignals

}

}

object UssStore {

case class Query(

userId: UserId,

params: configapi.Params,

product: Product)

val EnabledNegativeSourceTypes: Set[SourceType] =

Set(SourceType.AccountBlock, SourceType.AccountMute)

private val EnabledNegativeSignalTypes: Set[SignalType] =

Set(SignalType.AccountBlock, SignalType.AccountMute)

}