package com.twitter.visibility.rules

import com.twitter.timelines.configapi.Params

import com.twitter.visibility.common.ModelScoreThresholds

import com.twitter.visibility.configapi.configs.DeciderKey

import com.twitter.visibility.configapi.params.FSRuleParams.HighSpammyTweetContentScoreConvoDownrankAbusiveQualityThresholdParam

import com.twitter.visibility.configapi.params.RuleParams.EnableDownrankSpamReplySectioningRuleParam

import com.twitter.visibility.configapi.params.RuleParams.EnableNotGraduatedDownrankConvosAbusiveQualityRuleParam

import com.twitter.visibility.configapi.params.RuleParams.NotGraduatedUserLabelRuleHoldbackExperimentParam

import com.twitter.visibility.configapi.params.TimelineConversationsDownrankingSpecificParams.\_

import com.twitter.visibility.configapi.params.RuleParam

import com.twitter.visibility.models.TweetSafetyLabelType

import com.twitter.visibility.models.UserLabelValue

import com.twitter.visibility.rules.Condition.\_

import com.twitter.visibility.rules.RuleActionSourceBuilder.TweetSafetyLabelSourceBuilder

import com.twitter.visibility.rules.RuleActionSourceBuilder.UserSafetyLabelSourceBuilder

object DownrankingRules {

val ToxicityScoreAboveDownrankAbusiveQualitySectionThresholdCondition: TweetHasLabelWithLanguageScoreAboveThreshold =

TweetHasLabelWithLanguageScoreAboveThreshold(

safetyLabel = TweetSafetyLabelType.HighToxicityScore,

languagesToScoreThresholds = ModelScoreThresholds.ToxicityAbusiveQualityLanguagesToThresholds

)

val ToxicityScoreAboveDownrankLowQualitySectionThresholdCondition: TweetHasLabelWithLanguageScoreAboveThreshold =

TweetHasLabelWithLanguageScoreAboveThreshold(

safetyLabel = TweetSafetyLabelType.HighToxicityScore,

languagesToScoreThresholds = ModelScoreThresholds.ToxicityLowQualityLanguagesToThresholds

)

val ToxicityScoreAboveDownrankHighQualitySectionThresholdCondition: TweetHasLabelWithLanguageScoreAboveThreshold =

TweetHasLabelWithLanguageScoreAboveThreshold(

safetyLabel = TweetSafetyLabelType.HighToxicityScore,

languagesToScoreThresholds = ModelScoreThresholds.ToxicityHighQualityLanguagesToThresholds

)

val HighSpammyTweetContentScoreConvoDownrankAbusiveQualityCondition: Condition =

TweetHasLabelWithScoreAboveThresholdWithParam(

TweetSafetyLabelType.HighSpammyTweetContentScore,

HighSpammyTweetContentScoreConvoDownrankAbusiveQualityThresholdParam)

val HighCryptospamScoreConvoDownrankAbusiveQualityCondition: Condition =

TweetHasLabel(TweetSafetyLabelType.HighCryptospamScore)

}

object HighToxicityScoreDownrankHighQualitySectionRule

extends ConditionWithNotInnerCircleOfFriendsRule(

Downrank,

DownrankingRules.ToxicityScoreAboveDownrankHighQualitySectionThresholdCondition

)

with DoesLogVerdictDecidered {

override def actionSourceBuilder: Option[RuleActionSourceBuilder] = Some(

TweetSafetyLabelSourceBuilder(TweetSafetyLabelType.HighToxicityScore))

override def verdictLogDeciderKey = DeciderKey.EnableDownlevelRuleVerdictLogging

}

object HighToxicityScoreDownrankLowQualitySectionRule

extends ConditionWithNotInnerCircleOfFriendsRule(

ConversationSectionLowQuality,

DownrankingRules.ToxicityScoreAboveDownrankLowQualitySectionThresholdCondition

)

with DoesLogVerdict {

override def actionSourceBuilder: Option[RuleActionSourceBuilder] = Some(

TweetSafetyLabelSourceBuilder(TweetSafetyLabelType.HighToxicityScore))

}

object HighToxicityScoreDownrankAbusiveQualitySectionRule

extends ConditionWithNotInnerCircleOfFriendsRule(

ConversationSectionAbusiveQuality,

DownrankingRules.ToxicityScoreAboveDownrankAbusiveQualitySectionThresholdCondition

)

with DoesLogVerdict {

override def actionSourceBuilder: Option[RuleActionSourceBuilder] = Some(

TweetSafetyLabelSourceBuilder(TweetSafetyLabelType.HighToxicityScore))

}

object UntrustedUrlConversationsTweetLabelRule

extends ConditionWithNotInnerCircleOfFriendsRule(

ConversationSectionAbusiveQuality,

TweetHasLabel(TweetSafetyLabelType.UntrustedUrl)

)

with DoesLogVerdictDecidered {

override def actionSourceBuilder: Option[RuleActionSourceBuilder] = Some(

TweetSafetyLabelSourceBuilder(TweetSafetyLabelType.UntrustedUrl))

override def verdictLogDeciderKey = DeciderKey.EnableDownlevelRuleVerdictLogging

}

object DownrankSpamReplyConversationsTweetLabelRule

extends ConditionWithNotInnerCircleOfFriendsRule(

ConversationSectionAbusiveQuality,

TweetHasLabel(TweetSafetyLabelType.DownrankSpamReply)

)

with DoesLogVerdictDecidered {

override def enabled: Seq[RuleParam[Boolean]] =

Seq(EnableDownrankSpamReplySectioningRuleParam)

override def actionSourceBuilder: Option[RuleActionSourceBuilder] = Some(

TweetSafetyLabelSourceBuilder(TweetSafetyLabelType.DownrankSpamReply))

override def verdictLogDeciderKey = DeciderKey.EnableDownlevelRuleVerdictLogging

}

object DownrankSpamReplyConversationsAuthorLabelRule

extends AuthorLabelWithNotInnerCircleOfFriendsRule(

ConversationSectionAbusiveQuality,

UserLabelValue.DownrankSpamReply

)

with DoesLogVerdictDecidered {

override def enabled: Seq[RuleParam[Boolean]] =

Seq(EnableDownrankSpamReplySectioningRuleParam)

override def verdictLogDeciderKey = DeciderKey.EnableDownlevelRuleVerdictLogging

override def actionSourceBuilder: Option[RuleActionSourceBuilder] = Some(

UserSafetyLabelSourceBuilder(UserLabelValue.DownrankSpamReply))

}

object NotGraduatedConversationsAuthorLabelRule

extends AuthorLabelWithNotInnerCircleOfFriendsRule(

ConversationSectionLowQuality,

UserLabelValue.NotGraduated

)

with DoesLogVerdictDecidered {

override def enabled: Seq[RuleParam[Boolean]] =

Seq(EnableNotGraduatedDownrankConvosAbusiveQualityRuleParam)

override def holdbacks: Seq[RuleParam[Boolean]] = Seq(

NotGraduatedUserLabelRuleHoldbackExperimentParam)

override def verdictLogDeciderKey = DeciderKey.EnableDownlevelRuleVerdictLogging

override def actionSourceBuilder: Option[RuleActionSourceBuilder] = Some(

UserSafetyLabelSourceBuilder(UserLabelValue.NotGraduated))

}

object HighProactiveTosScoreTweetLabelDownrankingRule

extends ConditionWithNotInnerCircleOfFriendsRule(

ConversationSectionAbusiveQuality,

TweetHasLabel(TweetSafetyLabelType.HighProactiveTosScore)

)

with DoesLogVerdictDecidered {

override def actionSourceBuilder: Option[RuleActionSourceBuilder] = Some(

TweetSafetyLabelSourceBuilder(TweetSafetyLabelType.HighProactiveTosScore))

override def verdictLogDeciderKey = DeciderKey.EnableDownlevelRuleVerdictLogging

}

object HighPSpammyTweetScoreDownrankLowQualitySectionRule

extends ConditionWithNotInnerCircleOfFriendsRule(

action = ConversationSectionLowQuality,

condition = TweetHasLabelWithScoreAboveThreshold(

TweetSafetyLabelType.HighPSpammyTweetScore,

ModelScoreThresholds.HighPSpammyTweetScoreThreshold)

)

with DoesLogVerdictDecidered {

override def enabled: Seq[RuleParam[Boolean]] = Seq(

EnablePSpammyTweetDownrankConvosLowQualityParam)

override def actionSourceBuilder: Option[RuleActionSourceBuilder] = Some(

TweetSafetyLabelSourceBuilder(TweetSafetyLabelType.HighPSpammyTweetScore))

override def verdictLogDeciderKey: DeciderKey.Value =

DeciderKey.EnableSpammyTweetRuleVerdictLogging

}

object HighSpammyTweetContentScoreConvoDownrankAbusiveQualityRule

extends ConditionWithNotInnerCircleOfFriendsRule(

action = ConversationSectionAbusiveQuality,

condition = And(

Not(IsTweetInTweetLevelStcmHoldback),

DownrankingRules.HighSpammyTweetContentScoreConvoDownrankAbusiveQualityCondition)

)

with DoesLogVerdictDecidered {

override def isEnabled(params: Params): Boolean = {

params(EnableHighSpammyTweetContentScoreConvoDownrankAbusiveQualityRuleParam)

}

override def actionSourceBuilder: Option[RuleActionSourceBuilder] = Some(

TweetSafetyLabelSourceBuilder(TweetSafetyLabelType.HighSpammyTweetContentScore))

override def verdictLogDeciderKey = DeciderKey.EnableDownlevelRuleVerdictLogging

}

object HighCryptospamScoreConvoDownrankAbusiveQualityRule

extends ConditionWithNotInnerCircleOfFriendsRule(

action = ConversationSectionAbusiveQuality,

condition = DownrankingRules.HighCryptospamScoreConvoDownrankAbusiveQualityCondition

)

with DoesLogVerdictDecidered {

override def isEnabled(params: Params): Boolean = {

params(EnableHighCryptospamScoreConvoDownrankAbusiveQualityRuleParam)

}

override def actionSourceBuilder: Option[RuleActionSourceBuilder] = Some(

TweetSafetyLabelSourceBuilder(TweetSafetyLabelType.HighCryptospamScore))

override def verdictLogDeciderKey = DeciderKey.EnableDownlevelRuleVerdictLogging

}

object RitoActionedTweetDownrankLowQualitySectionRule

extends ConditionWithNotInnerCircleOfFriendsRule(

action = ConversationSectionLowQuality,

condition = TweetHasLabel(TweetSafetyLabelType.RitoActionedTweet)

)

with DoesLogVerdictDecidered {

override def enabled: Seq[RuleParam[Boolean]] = Seq(

EnableRitoActionedTweetDownrankConvosLowQualityParam)

override def actionSourceBuilder: Option[RuleActionSourceBuilder] = Some(

TweetSafetyLabelSourceBuilder(TweetSafetyLabelType.RitoActionedTweet))

override def verdictLogDeciderKey = DeciderKey.EnableDownlevelRuleVerdictLogging

}