package com.twitter.frigate.pushservice.predicate.ntab\_caret\_fatigue

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.frigate.common.predicate.FatiguePredicate

import com.twitter.frigate.pushservice.model.PushTypes.PushCandidate

import com.twitter.hermit.predicate.NamedPredicate

import com.twitter.frigate.common.base.Candidate

import com.twitter.frigate.common.base.TargetInfo

import com.twitter.frigate.common.rec\_types.RecTypes

import com.twitter.frigate.common.base.{RecommendationType => BaseRecommendationType}

import com.twitter.frigate.common.predicate.CandidateWithRecommendationTypeAndTargetInfoWithCaretFeedbackHistory

import com.twitter.frigate.common.predicate.FrigateHistoryFatiguePredicate.TimeSeries

import com.twitter.notificationservice.thriftscala.CaretFeedbackDetails

import com.twitter.frigate.pushservice.model.PushTypes.Target

import com.twitter.frigate.pushservice.predicate.CaretFeedbackHistoryFilter

object NtabCaretClickContFnFatiguePredicate {

private val MagicRecsCategory = "MagicRecs"

def ntabCaretClickContFnFatiguePredicates(

filterHistory: TimeSeries => TimeSeries =

FatiguePredicate.recTypesOnlyFilter(RecTypes.sharedNTabCaretFatigueTypes),

filterCaretFeedbackHistory: Target => Seq[

CaretFeedbackDetails

] => Seq[CaretFeedbackDetails] =

CaretFeedbackHistoryFilter.caretFeedbackHistoryFilter(Seq(MagicRecsCategory)),

filterInlineFeedbackHistory: Seq[FeedbackModel] => Seq[FeedbackModel] =

NtabCaretClickFatigueUtils.feedbackModelFilterByCRT(RecTypes.sharedNTabCaretFatigueTypes),

name: String = "NTabCaretClickFnCandidatePredicates"

)(

implicit globalStats: StatsReceiver

): NamedPredicate[PushCandidate] = {

val scopedStats = globalStats.scope(name)

CRTBasedNtabCaretClickFatiguePredicates

.f1TriggeredCRTBasedNtabCaretClickFnFatiguePredicate[

Candidate with BaseRecommendationType with TargetInfo[

Target

]

](

filterHistory = filterHistory,

filterCaretFeedbackHistory = filterCaretFeedbackHistory,

filterInlineFeedbackHistory = filterInlineFeedbackHistory

)

.applyOnlyToCandidateWithRecommendationTypeAndTargetWithCaretFeedbackHistory

.withName("f1\_triggered\_fn\_seelessoften\_fatigue")

.andThen(

CRTBasedNtabCaretClickFatiguePredicates

.nonF1TriggeredCRTBasedNtabCaretClickFnFatiguePredicate[

Candidate with BaseRecommendationType with TargetInfo[

Target

]

](

filterHistory = filterHistory,

filterCaretFeedbackHistory = filterCaretFeedbackHistory,

filterInlineFeedbackHistory = filterInlineFeedbackHistory

)

.applyOnlyToCandidateWithRecommendationTypeAndTargetWithCaretFeedbackHistory)

.withName("nonf1\_triggered\_fn\_seelessoften\_fatigue")

.andThen(

CRTBasedNtabCaretClickFatiguePredicates

.tripHqTweetTriggeredCRTBasedNtabCaretClickFnFatiguePredicate[

Candidate with BaseRecommendationType with TargetInfo[

Target

]

](

filterHistory = filterHistory,

filterCaretFeedbackHistory = filterCaretFeedbackHistory,

filterInlineFeedbackHistory = filterInlineFeedbackHistory

)

.applyOnlyToCandidateWithRecommendationTypeAndTargetWithCaretFeedbackHistory)

.withName("trip\_hq\_tweet\_triggered\_fn\_seelessoften\_fatigue")

.andThen(

CRTBasedNtabCaretClickFatiguePredicates

.genericCRTBasedNtabCaretClickFnFatiguePredicate[

Candidate with BaseRecommendationType with TargetInfo[

Target

]

](

filterHistory = filterHistory,

filterCaretFeedbackHistory = filterCaretFeedbackHistory,

filterInlineFeedbackHistory = filterInlineFeedbackHistory)

.applyOnlyToCandidateWithRecommendationTypeAndTargetWithCaretFeedbackHistory

.withName("generic\_fn\_seelessoften\_fatigue")

)

}

}