package com.twitter.visibility.interfaces.tweets.enrichments

import com.twitter.visibility.builder.VisibilityResult

import com.twitter.visibility.builder.tweets.TweetVisibilityNudgeSourceWrapper

import com.twitter.visibility.common.actions.TweetVisibilityNudgeReason.SemanticCoreMisinformationLabelReason

import com.twitter.visibility.rules.Action

import com.twitter.visibility.rules.LocalizedNudge

import com.twitter.visibility.rules.SoftIntervention

import com.twitter.visibility.rules.TweetVisibilityNudge

object TweetVisibilityNudgeEnrichment {

def apply(

result: VisibilityResult,

tweetVisibilityNudgeSourceWrapper: TweetVisibilityNudgeSourceWrapper,

languageCode: String,

countryCode: Option[String]

): VisibilityResult = {

val softIntervention = extractSoftIntervention(result.verdict, result.secondaryVerdicts)

val enrichedPrimaryVerdict = enrichAction(

result.verdict,

tweetVisibilityNudgeSourceWrapper,

softIntervention,

languageCode,

countryCode)

val enrichedSecondaryVerdicts: Seq[Action] =

result.secondaryVerdicts.map(sv =>

enrichAction(

sv,

tweetVisibilityNudgeSourceWrapper,

softIntervention,

languageCode,

countryCode))

result.copy(verdict = enrichedPrimaryVerdict, secondaryVerdicts = enrichedSecondaryVerdicts)

}

private def extractSoftIntervention(

primary: Action,

secondaries: Seq[Action]

): Option[SoftIntervention] = {

primary match {

case si: SoftIntervention => Some(si)

case \_ =>

secondaries.collectFirst {

case sv: SoftIntervention => sv

}

}

}

private def enrichAction(

action: Action,

tweetVisibilityNudgeSourceWrapper: TweetVisibilityNudgeSourceWrapper,

softIntervention: Option[SoftIntervention],

languageCode: String,

countryCode: Option[String]

): Action = {

action match {

case TweetVisibilityNudge(reason, None) =>

val localizedNudge =

tweetVisibilityNudgeSourceWrapper.getLocalizedNudge(reason, languageCode, countryCode)

if (reason == SemanticCoreMisinformationLabelReason)

TweetVisibilityNudge(

reason,

enrichLocalizedMisInfoNudge(localizedNudge, softIntervention))

else

TweetVisibilityNudge(reason, localizedNudge)

case \_ => action

}

}

private def enrichLocalizedMisInfoNudge(

localizedNudge: Option[LocalizedNudge],

softIntervention: Option[SoftIntervention]

): Option[LocalizedNudge] = {

softIntervention match {

case Some(si) => {

val enrichedLocalizedNudge = localizedNudge.map { ln =>

val enrichedLocalizedNudgeActions = ln.localizedNudgeActions.map { na =>

val enrichedPayload = na.nudgeActionPayload.map { payload =>

payload.copy(ctaUrl = si.detailsUrl, heading = si.warning)

}

na.copy(nudgeActionPayload = enrichedPayload)

}

ln.copy(localizedNudgeActions = enrichedLocalizedNudgeActions)

}

enrichedLocalizedNudge

}

case None => localizedNudge

}

}

}