package com.twitter.frigate.pushservice.predicate

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.frigate.common.base.\_

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

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

import com.twitter.frigate.pushservice.params.PushFeatureSwitchParams

import com.twitter.frigate.pushservice.util.CandidateUtil

import com.twitter.hermit.predicate.NamedPredicate

import com.twitter.hermit.predicate.Predicate

import com.twitter.language.normalization.UserDisplayLanguage

import com.twitter.util.Future

object TweetLanguagePredicate {

def oonTweeetLanguageMatch(

)(

implicit stats: StatsReceiver

): NamedPredicate[

PushCandidate with RecommendationType with TweetDetails

] = {

val name = "oon\_tweet\_language\_predicate"

val scopedStatsReceiver = stats.scope(name)

val oonCandidatesCounter =

scopedStatsReceiver.counter("oon\_candidates")

val enableFilterCounter =

scopedStatsReceiver.counter("enabled\_filter")

val skipMediaTweetsCounter =

scopedStatsReceiver.counter("skip\_media\_tweets")

Predicate

.fromAsync { candidate: PushCandidate with RecommendationType with TweetDetails =>

val target = candidate.target

val crt = candidate.commonRecType

val isOonCandidate = RecTypes.isOutOfNetworkTweetRecType(crt) ||

RecTypes.outOfNetworkTopicTweetTypes.contains(crt)

if (CandidateUtil.shouldApplyHealthQualityFilters(candidate) && isOonCandidate) {

oonCandidatesCounter.incr()

target.featureMap.map { featureMap =>

val userPreferredLanguages = featureMap.sparseBinaryFeatures

.getOrElse("user.language.user.preferred\_contents", Set.empty[String])

val userEngagementLanguages = featureMap.sparseContinuousFeatures.getOrElse(

"user.language.user.engagements",

Map.empty[String, Double])

val userFollowLanguages = featureMap.sparseContinuousFeatures.getOrElse(

"user.language.user.following\_accounts",

Map.empty[String, Double])

val userProducedTweetLanguages = featureMap.sparseContinuousFeatures

.getOrElse("user.language.user.produced\_tweets", Map.empty)

val userDeviceLanguages = featureMap.sparseContinuousFeatures.getOrElse(

"user.language.user.recent\_devices",

Map.empty[String, Double])

val tweetLanguageOpt = candidate.categoricalFeatures

.get(target.params(PushFeatureSwitchParams.TweetLanguageFeatureNameParam))

if (userPreferredLanguages.isEmpty)

scopedStatsReceiver.counter("userPreferredLanguages\_empty").incr()

if (userEngagementLanguages.isEmpty)

scopedStatsReceiver.counter("userEngagementLanguages\_empty").incr()

if (userFollowLanguages.isEmpty)

scopedStatsReceiver.counter("userFollowLanguages\_empty").incr()

if (userProducedTweetLanguages.isEmpty)

scopedStatsReceiver

.counter("userProducedTweetLanguages\_empty")

.incr()

if (userDeviceLanguages.isEmpty)

scopedStatsReceiver.counter("userDeviceLanguages\_empty").incr()

if (tweetLanguageOpt.isEmpty) scopedStatsReceiver.counter("tweetLanguage\_empty").incr()

val tweetLanguage = tweetLanguageOpt.getOrElse("und")

val undefinedTweetLanguages = Set("")

if (!undefinedTweetLanguages.contains(tweetLanguage)) {

lazy val userInferredLanguageThreshold =

target.params(PushFeatureSwitchParams.UserInferredLanguageThresholdParam)

lazy val userDeviceLanguageThreshold =

target.params(PushFeatureSwitchParams.UserDeviceLanguageThresholdParam)

lazy val enableTweetLanguageFilter =

target.params(PushFeatureSwitchParams.EnableTweetLanguageFilter)

lazy val skipLanguageFilterForMediaTweets =

target.params(PushFeatureSwitchParams.SkipLanguageFilterForMediaTweets)

lazy val allLanguages = userPreferredLanguages ++

userEngagementLanguages.filter(\_.\_2 > userInferredLanguageThreshold).keySet ++

userFollowLanguages.filter(\_.\_2 > userInferredLanguageThreshold).keySet ++

userProducedTweetLanguages.filter(\_.\_2 > userInferredLanguageThreshold).keySet ++

userDeviceLanguages.filter(\_.\_2 > userDeviceLanguageThreshold).keySet

if (enableTweetLanguageFilter && allLanguages.nonEmpty) {

enableFilterCounter.incr()

val hasMedia = candidate.hasPhoto || candidate.hasVideo

if (hasMedia && skipLanguageFilterForMediaTweets) {

skipMediaTweetsCounter.incr()

true

} else {

allLanguages.map(UserDisplayLanguage.toTweetLanguage).contains(tweetLanguage)

}

} else true

} else true

}

} else Future.True

}

.withStats(stats.scope(name))

.withName(name)

}

}