package com.twitter.cr\_mixer.filter

import com.twitter.cr\_mixer.filter.VideoTweetFilter.FilterConfig

import com.twitter.cr\_mixer.model.CandidateGeneratorQuery

import com.twitter.cr\_mixer.model.CrCandidateGeneratorQuery

import com.twitter.cr\_mixer.model.InitialCandidate

import com.twitter.cr\_mixer.model.RelatedTweetCandidateGeneratorQuery

import com.twitter.cr\_mixer.model.RelatedVideoTweetCandidateGeneratorQuery

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

import com.twitter.util.Future

import javax.inject.Singleton

@Singleton

case class VideoTweetFilter() extends FilterBase {

override val name: String = this.getClass.getCanonicalName

override type ConfigType = FilterConfig

override def filter(

candidates: Seq[Seq[InitialCandidate]],

config: ConfigType

): Future[Seq[Seq[InitialCandidate]]] = {

Future.value(candidates.map {

\_.flatMap {

candidate =>

if (!config.enableVideoTweetFilter) {

Some(candidate)

} else {

// if hasVideo is true, hasImage, hasGif should be false

val hasVideo = checkTweetInfoAttribute(candidate.tweetInfo.hasVideo)

val isHighMediaResolution =

checkTweetInfoAttribute(candidate.tweetInfo.isHighMediaResolution)

val isQuoteTweet = checkTweetInfoAttribute(candidate.tweetInfo.isQuoteTweet)

val isReply = checkTweetInfoAttribute(candidate.tweetInfo.isReply)

val hasMultipleMedia = checkTweetInfoAttribute(candidate.tweetInfo.hasMultipleMedia)

val hasUrl = checkTweetInfoAttribute(candidate.tweetInfo.hasUrl)

if (hasVideo && isHighMediaResolution && !isQuoteTweet &&

!isReply && !hasMultipleMedia && !hasUrl) {

Some(candidate)

} else {

None

}

}

}

})

}

def checkTweetInfoAttribute(attributeOpt: => Option[Boolean]): Boolean = {

if (attributeOpt.isDefined)

attributeOpt.get

else {

// takes Quoted Tweet (TweetInfo.isQuoteTweet) as an example,

// if the attributeOpt is None, we by default say it is not a quoted tweet

// similarly, if TweetInfo.hasVideo is a None,

// we say it does not have video.

false

}

}

override def requestToConfig[CGQueryType <: CandidateGeneratorQuery](

query: CGQueryType

): FilterConfig = {

val enableVideoTweetFilter = query match {

case \_: CrCandidateGeneratorQuery | \_: RelatedTweetCandidateGeneratorQuery |

\_: RelatedVideoTweetCandidateGeneratorQuery =>

query.params(VideoTweetFilterParams.EnableVideoTweetFilterParam)

case \_ => false // e.g., GetRelatedTweets()

}

FilterConfig(

enableVideoTweetFilter = enableVideoTweetFilter

)

}

}

object VideoTweetFilter {

// extend the filterConfig to add more flags if needed.

// now they are hardcoded according to the prod setting

case class FilterConfig(

enableVideoTweetFilter: Boolean)

}