package com.twitter.timelineranker.util

import com.twitter.timelines.model.TweetId

import com.twitter.timelines.model.UserId

import com.twitter.timelines.model.tweet.HydratedTweet

object ExtendedRepliesFilter {

private[util] def isExtendedReply(tweet: HydratedTweet, followedUserIds: Seq[UserId]): Boolean = {

tweet.hasReply &&

tweet.directedAtUser.exists(!followedUserIds.contains(\_)) &&

followedUserIds.contains(tweet.userId)

}

private[util] def isNotQualifiedExtendedReply(

tweet: HydratedTweet,

userId: UserId,

followedUserIds: Seq[UserId],

mutedUserIds: Set[UserId],

sourceTweetsById: Map[TweetId, HydratedTweet]

): Boolean = {

val currentUserId = userId

isExtendedReply(tweet, followedUserIds) &&

!(

!tweet.isRetweet &&

// and the extended reply must be directed at someone other than the current user

tweet.directedAtUser.exists(\_ != currentUserId) &&

// There must be a source tweet

tweet.inReplyToTweetId

.flatMap(sourceTweetsById.get)

.filter { c =>

// and the author of the source tweet must be non zero

(c.userId != 0) &&

(c.userId != currentUserId) && // and not by the current user

(!c.hasReply) && // and a root tweet, i.e. not a reply

(!c.isRetweet) && // and not a retweet

(c.userId != tweet.userId) // and not a by the same user

}

// and not by a muted user

.exists(sourceTweet => !mutedUserIds.contains(sourceTweet.userId))

)

}

private[util] def isNotValidExpandedExtendedReply(

tweet: HydratedTweet,

viewingUserId: UserId,

followedUserIds: Seq[UserId],

mutedUserIds: Set[UserId],

sourceTweetsById: Map[TweetId, HydratedTweet]

): Boolean = {

// An extended reply is valid if we hydrated the in-reply to tweet

val isValidExtendedReply =

!tweet.isRetweet && // extended replies must be source tweets

tweet.directedAtUser.exists(

\_ != viewingUserId) && // the extended reply must be directed at someone other than the viewing user

tweet.inReplyToTweetId

.flatMap(

sourceTweetsById.get

) // there must be an in-reply-to tweet matching the following properities

.exists { inReplyToTweet =>

(inReplyToTweet.userId > 0) && // and the in-reply to author is valid

(inReplyToTweet.userId != viewingUserId) && // the reply can not be in reply to the viewing user's tweet

!inReplyToTweet.isRetweet && // and the in-reply-to tweet is not a retweet (this should always be true?)

!mutedUserIds.contains(

inReplyToTweet.userId) && // and the in-reply-to user is not muted

inReplyToTweet.inReplyToUserId.forall(r =>

!mutedUserIds

.contains(r)) // if there is an in-reply-to-in-reply-to user they are not muted

}

// filter any invalid extended reply

isExtendedReply(tweet, followedUserIds) && !isValidExtendedReply

}

}