package com.twitter.timelineranker.parameters.recap

import com.twitter.timelineranker.model.RecapQuery

import com.twitter.timelines.util.bounds.BoundsWithDefault

object RecapQueryContext {

val MaxFollowedUsers: BoundsWithDefault[Int] = BoundsWithDefault[Int](1, 3000, 1000)

val MaxCountMultiplier: BoundsWithDefault[Double] = BoundsWithDefault[Double](0.1, 2.0, 2.0)

val MaxRealGraphAndFollowedUsers: BoundsWithDefault[Int] = BoundsWithDefault[Int](0, 2000, 1000)

def getDefaultContext(query: RecapQuery): RecapQueryContext = {

new RecapQueryContextImpl(

query,

getEnableHydrationUsingTweetyPie = () => false,

getMaxFollowedUsers = () => MaxFollowedUsers.default,

getMaxCountMultiplier = () => MaxCountMultiplier.default,

getEnableRealGraphUsers = () => false,

getOnlyRealGraphUsers = () => false,

getMaxRealGraphAndFollowedUsers = () => MaxRealGraphAndFollowedUsers.default,

getEnableTextFeatureHydration = () => false

)

}

}

// Note that methods that return parameter value always use () to indicate that

// side effects may be involved in their invocation.

trait RecapQueryContext {

def query: RecapQuery

// If true, tweet hydration are performed by calling TweetyPie.

// Otherwise, tweets are partially hydrated based on information in ThriftSearchResult.

def enableHydrationUsingTweetyPie(): Boolean

// Maximum number of followed user accounts to use when fetching recap tweets.

def maxFollowedUsers(): Int

// We multiply maxCount (caller supplied value) by this multiplier and fetch those many

// candidates from search so that we are left with sufficient number of candidates after

// hydration and filtering.

def maxCountMultiplier(): Double

// Only used if user follows >= 1000.

// If true, fetches recap/recycled tweets using author seedset mixing with real graph users and followed users.

// Otherwise, fetches recap/recycled tweets only using followed users

def enableRealGraphUsers(): Boolean

// Only used if enableRealGraphUsers is true.

// If true, user seedset only contains real graph users.

// Otherwise, user seedset contains real graph users and recent followed users.

def onlyRealGraphUsers(): Boolean

// Only used if enableRealGraphUsers is true and onlyRealGraphUsers is false.

// Maximum number of real graph users and recent followed users when mixing recent/real-graph users.

def maxRealGraphAndFollowedUsers(): Int

// If true, text features are hydrated for prediction.

// Otherwise those feature values are not set at all.

def enableTextFeatureHydration(): Boolean

}

class RecapQueryContextImpl(

override val query: RecapQuery,

getEnableHydrationUsingTweetyPie: () => Boolean,

getMaxFollowedUsers: () => Int,

getMaxCountMultiplier: () => Double,

getEnableRealGraphUsers: () => Boolean,

getOnlyRealGraphUsers: () => Boolean,

getMaxRealGraphAndFollowedUsers: () => Int,

getEnableTextFeatureHydration: () => Boolean)

extends RecapQueryContext {

override def enableHydrationUsingTweetyPie(): Boolean = { getEnableHydrationUsingTweetyPie() }

override def maxFollowedUsers(): Int = { getMaxFollowedUsers() }

override def maxCountMultiplier(): Double = { getMaxCountMultiplier() }

override def enableRealGraphUsers(): Boolean = { getEnableRealGraphUsers() }

override def onlyRealGraphUsers(): Boolean = { getOnlyRealGraphUsers() }

override def maxRealGraphAndFollowedUsers(): Int = { getMaxRealGraphAndFollowedUsers() }

override def enableTextFeatureHydration(): Boolean = { getEnableTextFeatureHydration() }

}