package com.twitter.timelineranker.parameters.recap

import com.twitter.timelines.configapi.decider.\_

import com.twitter.timelines.configapi.FSBoundedParam

import com.twitter.timelines.configapi.FSParam

import com.twitter.timelines.configapi.Param

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

object RecapParams {

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)

val ProbabilityRandomTweet: BoundsWithDefault[Double] = BoundsWithDefault[Double](0.0, 1.0, 0.0)

/\*\*

\* Controls limit on the number of followed users fetched from SGS.

\*

\* The specific default value below is for blender-timelines parity.

\*/

object MaxFollowedUsersParam

extends FSBoundedParam[Int](

name = "recap\_max\_followed\_users",

default = MaxFollowedUsers.default,

min = MaxFollowedUsers.bounds.minInclusive,

max = MaxFollowedUsers.bounds.maxInclusive

)

/\*\*

\* Controls limit on the number of hits for Earlybird.

\* We added it solely for backward compatibility, to align with recycled.

\* RecapSource is deprecated, but, this param is used by RecapAuthor source

\*/

object RelevanceOptionsMaxHitsToProcessParam

extends FSBoundedParam[Int](

name = "recap\_relevance\_options\_max\_hits\_to\_process",

default = 500,

min = 100,

max = 20000

)

/\*\*

\* Enables fetching author seedset from real graph users. Only used if user follows >= 1000.

\* If true, expands author seedset with real graph users and recent followed users.

\* Otherwise, user seedset only includes followed users.

\*/

object EnableRealGraphUsersParam extends Param(false)

/\*\*

\* Only used if EnableRealGraphUsersParam is true and OnlyRealGraphUsersParam is false.

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

\*/

object MaxRealGraphAndFollowedUsersParam

extends Param(MaxRealGraphAndFollowedUsers.default)

with DeciderValueConverter[Int] {

override def convert: IntConverter[Int] =

OutputBoundIntConverter(MaxRealGraphAndFollowedUsers.bounds)

}

/\*\*

\* FS-controlled param to override the MaxRealGraphAndFollowedUsersParam decider value for experiments

\*/

object MaxRealGraphAndFollowedUsersFSOverrideParam

extends FSBoundedParam[Option[Int]](

name = "max\_real\_graph\_and\_followers\_users\_fs\_override\_param",

default = None,

min = Some(100),

max = Some(10000)

)

/\*\*

\* Experimental params for leveling the playing field between user folowees received from

\* real-graph and follow-graph stores.

\* Author relevance scores returned by real-graph are currently being used for light-ranking

\* in-network tweet candidates.

\* Follow-graph store returns the most recent followees without any relevance scores

\* We are trying to impute the missing scores by using aggregated statistics (min, avg, p50, etc.)

\* of real-graph scores.

\*/

object ImputeRealGraphAuthorWeightsParam

extends FSParam(name = "impute\_real\_graph\_author\_weights", default = false)

object ImputeRealGraphAuthorWeightsPercentileParam

extends FSBoundedParam[Int](

name = "impute\_real\_graph\_author\_weights\_percentile",

default = 50,

min = 0,

max = 99)

/\*\*

\* Enable running the new pipeline for recap author source

\*/

object EnableNewRecapAuthorPipeline extends Param(false)

/\*\*

\* Fallback value for maximum number of search results, if not specified by query.maxCount

\*/

object DefaultMaxTweetCount extends Param(200)

/\*\*

\* 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.

\*/

object MaxCountMultiplierParam

extends Param(MaxCountMultiplier.default)

with DeciderValueConverter[Double] {

override def convert: IntConverter[Double] =

OutputBoundIntConverter[Double](divideDeciderBy100 \_, MaxCountMultiplier.bounds)

}

/\*\*

\* Enables return all results from search index.

\*/

object EnableReturnAllResultsParam

extends FSParam(name = "recap\_enable\_return\_all\_results", default = false)

/\*\*

\* Includes one or multiple random tweets in the response.

\*/

object IncludeRandomTweetParam

extends FSParam(name = "recap\_include\_random\_tweet", default = false)

/\*\*

\* One single random tweet (true) or tag tweet as random with given probability (false).

\*/

object IncludeSingleRandomTweetParam

extends FSParam(name = "recap\_include\_random\_tweet\_single", default = true)

/\*\*

\* Probability to tag a tweet as random (will not be ranked).

\*/

object ProbabilityRandomTweetParam

extends FSBoundedParam(

name = "recap\_include\_random\_tweet\_probability",

default = ProbabilityRandomTweet.default,

min = ProbabilityRandomTweet.bounds.minInclusive,

max = ProbabilityRandomTweet.bounds.maxInclusive)

/\*\*

\* Enable extra sorting by score for search results.

\*/

object EnableExtraSortingInSearchResultParam extends Param(true)

/\*\*

\* Enables semantic core, penguin, and tweetypie content features in recap source.

\*/

object EnableContentFeaturesHydrationParam extends Param(true)

/\*\*

\* additionally enables tokens when hydrating content features.

\*/

object EnableTokensInContentFeaturesHydrationParam

extends FSParam(

name = "recap\_enable\_tokens\_in\_content\_features\_hydration",

default = false

)

/\*\*

\* additionally enables tweet text when hydrating content features.

\* This only works if EnableContentFeaturesHydrationParam is set to true

\*/

object EnableTweetTextInContentFeaturesHydrationParam

extends FSParam(

name = "recap\_enable\_tweet\_text\_in\_content\_features\_hydration",

default = false

)

/\*\*

\* Enables hydrating in-network inReplyToTweet features

\*/

object EnableInNetworkInReplyToTweetFeaturesHydrationParam

extends FSParam(

name = "recap\_enable\_in\_network\_in\_reply\_to\_tweet\_features\_hydration",

default = false

)

/\*\*

\* Enables hydrating root tweet of in-network replies and extended replies

\*/

object EnableReplyRootTweetHydrationParam

extends FSParam(

name = "recap\_enable\_reply\_root\_tweet\_hydration",

default = false

)

/\*\*

\* Enable setting tweetTypes in search queries with TweetKindOption in RecapQuery

\*/

object EnableSettingTweetTypesWithTweetKindOption

extends FSParam(

name = "recap\_enable\_setting\_tweet\_types\_with\_tweet\_kind\_option",

default = false

)

/\*\*

\* Enable relevance search, otherwise recency search from earlybird.

\*/

object EnableRelevanceSearchParam

extends FSParam(

name = "recap\_enable\_relevance\_search",

default = true

)

object EnableExpandedExtendedRepliesFilterParam

extends FSParam(

name = "recap\_enable\_expanded\_extended\_replies\_filter",

default = false

)

/\*\*

\* additionally enables conversationControl when hydrating content features.

\* This only works if EnableContentFeaturesHydrationParam is set to true

\*/

object EnableConversationControlInContentFeaturesHydrationParam

extends FSParam(

name = "conversation\_control\_in\_content\_features\_hydration\_recap\_enable",

default = false

)

object EnableTweetMediaHydrationParam

extends FSParam(

name = "tweet\_media\_hydration\_recap\_enable",

default = false

)

object EnableExcludeSourceTweetIdsQueryParam

extends FSParam[Boolean](

name = "recap\_exclude\_source\_tweet\_ids\_query\_enable",

default = false

)

}