namespace java com.twitter.search.common.ranking.thriftjava

#@namespace scala com.twitter.search.common.ranking.thriftscala

#@namespace strato com.twitter.search.common.ranking

namespace py gen.twitter.search.common.ranking.ranking

struct ThriftLinearFeatureRankingParams {

// values below this will set the score to the minimal one

1: optional double min = -1e+100

// values above this will set the score to the minimal one

2: optional double max = 1e+100

3: optional double weight = 0

}(persisted='true')

struct ThriftAgeDecayRankingParams {

// the rate in which the score of older tweets decreases

1: optional double slope = 0.003

// the age, in minutes, where the age score of a tweet is half of the latest tweet

2: optional double halflife = 360.0

// the minimal age decay score a tweet will have

3: optional double base = 0.6

}(persisted='true')

enum ThriftScoringFunctionType {

LINEAR = 1,

MODEL\_BASED = 4,

TENSORFLOW\_BASED = 5,

// deprecated

TOPTWEETS = 2,

EXPERIMENTAL = 3,

}

// The struct to define a class that is to be dynamically loaded in earlybird for

// experimentation.

struct ThriftExperimentClass {

// the fully qualified class name.

1: required string name

// data source location (class/jar file) for this dynamic class on HDFS

2: optional string location

// parameters in key-value pairs for this experimental class

3: optional map<string, double> params

}(persisted='true')

// Deprecated!!

struct ThriftQueryEngagementParams {

// Rate Boosts: given a rate (usually a small fraction), the score will be multiplied by

// (1 + rate) ^ boost

// 0 mean no boost, negative numbers are dampens

1: optional double retweetRateBoost = 0

2: optional double replyRateBoost = 0

3: optional double faveRateBoost = 0

}(persisted='true')

struct ThriftHostQualityParams {

// Multiplier applied to host score, for tweets that have links.

// A multiplier of 0 means that this boost is not applied

1: optional double multiplier = 0.0

// Do not apply the multiplier to hosts with score above this level.

// If 0, the multiplier will be applied to any host.

2: optional double maxScoreToModify = 0.0

// Do not apply the multiplier to hosts with score below this level.

// If 0, the multiplier will be applied to any host.

3: optional double minScoreToModify = 0.0

// If true, score modification will be applied to hosts that have unknown scores.

// The host-score used will be lower than the score of any known host.

4: optional bool applyToUnknownHosts = 0

}(persisted='true')

struct ThriftCardRankingParams {

1: optional double hasCardBoost = 1.0

2: optional double domainMatchBoost = 1.0

3: optional double authorMatchBoost = 1.0

4: optional double titleMatchBoost = 1.0

5: optional double descriptionMatchBoost = 1.0

}(persisted='true')

# The ids are assigned in 'blocks'. For adding a new field, find an unused id in the appropriate

# block. Be sure to mention explicitly which ids have been removed so that they are not used again.

struct ThriftRankingParams {

1: optional ThriftScoringFunctionType type

// Dynamically loaded scorer and collector for quick experimentation.

40: optional ThriftExperimentClass expScorer

41: optional ThriftExperimentClass expCollector

// we must set it to a value that fits into a float: otherwise

// some earlybird classes that convert it to float will interpret

// it as Float.NEGATIVE\_INFINITY, and some comparisons will fail

2: optional double minScore = -1e+30

10: optional ThriftLinearFeatureRankingParams parusScoreParams

11: optional ThriftLinearFeatureRankingParams retweetCountParams

12: optional ThriftLinearFeatureRankingParams replyCountParams

15: optional ThriftLinearFeatureRankingParams reputationParams

16: optional ThriftLinearFeatureRankingParams luceneScoreParams

18: optional ThriftLinearFeatureRankingParams textScoreParams

19: optional ThriftLinearFeatureRankingParams urlParams

20: optional ThriftLinearFeatureRankingParams isReplyParams

21: optional ThriftLinearFeatureRankingParams directFollowRetweetCountParams

22: optional ThriftLinearFeatureRankingParams trustedCircleRetweetCountParams

23: optional ThriftLinearFeatureRankingParams favCountParams

24: optional ThriftLinearFeatureRankingParams multipleReplyCountParams

27: optional ThriftLinearFeatureRankingParams embedsImpressionCountParams

28: optional ThriftLinearFeatureRankingParams embedsUrlCountParams

29: optional ThriftLinearFeatureRankingParams videoViewCountParams

66: optional ThriftLinearFeatureRankingParams quotedCountParams

// A map from MutableFeatureType to linear ranking params

25: optional map<byte, ThriftLinearFeatureRankingParams> offlineExperimentalFeatureRankingParams

// if min/max for score or ThriftLinearFeatureRankingParams should always be

// applied or only to non-follows, non-self, non-verified

26: optional bool applyFiltersAlways = 0

// Whether to apply promotion/demotion at all for FeatureBasedScoringFunction

70: optional bool applyBoosts = 1

// UI language is english, tweet language is not

30: optional double langEnglishUIBoost = 0.3

// tweet language is english, UI language is not

31: optional double langEnglishTweetBoost = 0.7

// user language differs from tweet language, and neither is english

32: optional double langDefaultBoost = 0.1

// user that produced tweet is marked as spammer by metastore

33: optional double spamUserBoost = 1.0

// user that produced tweet is marked as nsfw by metastore

34: optional double nsfwUserBoost = 1.0

// user that produced tweet is marked as bot (self similarity) by metastore

35: optional double botUserBoost = 1.0

// An alternative way of using lucene score in the ranking function.

38: optional bool useLuceneScoreAsBoost = 0

39: optional double maxLuceneScoreBoost = 1.2

// Use user's consumed and produced languages for scoring

42: optional bool useUserLanguageInfo = 0

// Boost (demotion) if the tweet language is not one of user's understandable languages,

// nor interface language.

43: optional double unknownLanguageBoost = 0.01

// Use topic ids for scoring.

// Deprecated in SEARCH-8616.

44: optional bool deprecated\_useTopicIDsBoost = 0

// Parameters for topic id scoring. See TopicIDsBoostScorer (and its test) for details.

46: optional double deprecated\_maxTopicIDsBoost = 3.0

47: optional double deprecated\_topicIDsBoostExponent = 2.0;

48: optional double deprecated\_topicIDsBoostSlope = 2.0;

// Hit Attribute Demotion

60: optional bool enableHitDemotion = 0

61: optional double noTextHitDemotion = 1.0

62: optional double urlOnlyHitDemotion = 1.0

63: optional double nameOnlyHitDemotion = 1.0

64: optional double separateTextAndNameHitDemotion = 1.0

65: optional double separateTextAndUrlHitDemotion = 1.0

// multiplicative score boost for results deemed offensive

100: optional double offensiveBoost = 1

// multiplicative score boost for results in the searcher's social circle

101: optional double inTrustedCircleBoost = 1

// multiplicative score dampen for results with more than one hash tag

102: optional double multipleHashtagsOrTrendsBoost = 1

// multiplicative score boost for results in the searcher's direct follows

103: optional double inDirectFollowBoost = 1

// multiplicative score boost for results that has trends

104: optional double tweetHasTrendBoost = 1

// is tweet from verified account?

106: optional double tweetFromVerifiedAccountBoost = 1

// is tweet authored by the searcher? (boost is in addition to social boost)

107: optional double selfTweetBoost = 1

// multiplicative score boost for a tweet that has image url.

108: optional double tweetHasImageUrlBoost = 1

// multiplicative score boost for a tweet that has video url.

109: optional double tweetHasVideoUrlBoost = 1

// multiplicative score boost for a tweet that has news url.

110: optional double tweetHasNewsUrlBoost = 1

// is tweet from a blue-verified account?

111: optional double tweetFromBlueVerifiedAccountBoost = 1 (personalDataType = 'UserVerifiedFlag')

// subtractive penalty applied after boosts for out-of-network replies.

120: optional double outOfNetworkReplyPenalty = 10.0

150: optional ThriftQueryEngagementParams deprecatedQueryEngagementParams

160: optional ThriftHostQualityParams deprecatedHostQualityParams

// age decay params for regular tweets

203: optional ThriftAgeDecayRankingParams ageDecayParams

// for card ranking: map between card name ordinal (defined in com.twitter.search.common.constants.CardConstants)

// to ranking params

400: optional map<byte, ThriftCardRankingParams> cardRankingParams

// A map from tweet IDs to the score adjustment for that tweet. These are score

// adjustments that include one or more features that can depend on the query

// string. These features aren't indexed by Earlybird, and so their total contribution

// to the scoring function is passed in directly as part of the request. If present,

// the score adjustment for a tweet is directly added to the linear component of the

// scoring function. Since this signal can be made up of multiple features, any

// reweighting or combination of these features is assumed to be done by the caller

// (hence there is no need for a weight parameter -- the weights of the features

// included in this signal have already been incorporated by the caller).

151: optional map<i64, double> querySpecificScoreAdjustments

// A map from user ID to the score adjustment for tweets from that author.

// This field provides a way for adjusting the tweets of a specific set of users with a score

// that is not present in the Earlybird features but has to be passed from the clients, such as

// real graph weights or a combination of multiple features.

// This field should be used mainly for experimentation since it increases the size of the thrift

// requests.

154: optional map<i64, double> authorSpecificScoreAdjustments

// -------- Parameters for ThriftScoringFunctionType.MODEL\_BASED --------

// Selected models along with their weights for the linear combination

152: optional map<string, double> selectedModels

153: optional bool useLogitScore = false

// -------- Parameters for ThriftScoringFunctionType.TENSORFLOW\_BASED --------

// Selected tensorflow model

303: optional string selectedTensorflowModel

// -------- Deprecated Fields --------

// ID 303 has been used in the past. Resume additional deprecated fields from 304

105: optional double deprecatedTweetHasTrendInTrendingQueryBoost = 1

200: optional double deprecatedAgeDecaySlope = 0.003

201: optional double deprecatedAgeDecayHalflife = 360.0

202: optional double deprecatedAgeDecayBase = 0.6

204: optional ThriftAgeDecayRankingParams deprecatedAgeDecayForTrendsParams

301: optional double deprecatedNameQueryConfidence = 0.0

302: optional double deprecatedHashtagQueryConfidence = 0.0

// Whether to use old-style engagement features (normalized by LogNormalizer)

// or new ones (normalized by SingleBytePositiveFloatNormalizer)

50: optional bool useGranularEngagementFeatures = 0 // DEPRECATED!

}(persisted='true')

// This sorting mode is used by earlybird to retrieve the top-n facets that

// are returned to blender

enum ThriftFacetEarlybirdSortingMode {

SORT\_BY\_SIMPLE\_COUNT = 0,

SORT\_BY\_WEIGHTED\_COUNT = 1,

}

// This is the final sort order used by blender after all results from

// the earlybirds are merged

enum ThriftFacetFinalSortOrder {

// using the created\_at date of the first tweet that contained the facet

SCORE = 0,

SIMPLE\_COUNT = 1,

WEIGHTED\_COUNT = 2,

CREATED\_AT = 3

}

struct ThriftFacetRankingOptions {

// next available field ID = 38

// ======================================================================

// EARLYBIRD SETTINGS

//

// These parameters primarily affect how earlybird creates the top-k

// candidate list to be re-ranked by blender

// ======================================================================

// Dynamically loaded scorer and collector for quick experimentation.

26: optional ThriftExperimentClass expScorer

27: optional ThriftExperimentClass expCollector

// It should be less than or equal to reputationParams.min, and all

// tweepcreds between the two get a score of 1.0.

21: optional i32 minTweepcredFilterThreshold

// the maximum score a single tweet can contribute to the weightedCount

22: optional i32 maxScorePerTweet

15: optional ThriftFacetEarlybirdSortingMode sortingMode

// The number of top candidates earlybird returns to blender

16: optional i32 numCandidatesFromEarlybird = 100

// when to early terminate for facet search, overrides the setting in ThriftSearchQuery

34: optional i32 maxHitsToProcess = 1000

// for anti-gaming we want to limit the maximum amount of hits the same user can

// contribute. Set to -1 to disable the anti-gaming filter. Overrides the setting in

// ThriftSearchQuery

35: optional i32 maxHitsPerUser = 3

// if the tweepcred of the user is bigger than this value it will not be excluded

// by the anti-gaming filter. Overrides the setting in ThriftSearchQuery

36: optional i32 maxTweepcredForAntiGaming = 65

// these settings affect how earlybird computes the weightedCount

2: optional ThriftLinearFeatureRankingParams parusScoreParams

3: optional ThriftLinearFeatureRankingParams reputationParams

17: optional ThriftLinearFeatureRankingParams favoritesParams

33: optional ThriftLinearFeatureRankingParams repliesParams

37: optional map<byte, ThriftLinearFeatureRankingParams> rankingExpScoreParams

// penalty counter settings

6: optional i32 offensiveTweetPenalty // set to -1 to disable the offensive filter

7: optional i32 antigamingPenalty // set to -1 to disable antigaming filtering

// weight of penalty counts from all tweets containing a facet, not just the tweets

// matching the query

9: optional double queryIndependentPenaltyWeight // set to 0 to not use query independent penalty weights

// penalty for keyword stuffing

60: optional i32 multipleHashtagsOrTrendsPenalty

// Language related boosts, similar to those in relevance ranking options. By default they are

// all 1.0 (no-boost).

// When the user language is english, facet language is not

11: optional double langEnglishUIBoost = 1.0

// When the facet language is english, user language is not

12: optional double langEnglishFacetBoost = 1.0

// When the user language differs from facet/tweet language, and neither is english

13: optional double langDefaultBoost = 1.0

// ======================================================================

// BLENDER SETTINGS

//

// Settings for the facet relevance scoring happening in blender

// ======================================================================

// This block of parameters are only used in the FacetsFutureManager.

// limits to discard facets

// if a facet has a higher penalty count, it will not be returned

5: optional i32 maxPenaltyCount

// if a facet has a lower simple count, it will not be returned

28: optional i32 minSimpleCount

// if a facet has a lower weighted count, it will not be returned

8: optional i32 minCount

// the maximum allowed value for offensiveCount/facetCount a facet can have in order to be returned

10: optional double maxPenaltyCountRatio

// if set to true, then facets with offensive display tweets are excluded from the resultset

29: optional bool excludePossiblySensitiveFacets

// if set to true, then only facets that have a display tweet in their ThriftFacetCountMetadata object

// will be returned to the caller

30: optional bool onlyReturnFacetsWithDisplayTweet

// parameters for scoring force-inserted media items

// Please check FacetReRanker.java computeScoreForInserted() for their usage.

38: optional double forceInsertedBackgroundExp = 0.3

39: optional double forceInsertedMinBackgroundCount = 2

40: optional double forceInsertedMultiplier = 0.01

// -----------------------------------------------------

// weights for the facet ranking formula

18: optional double simpleCountWeight\_DEPRECATED

19: optional double weightedCountWeight\_DEPRECATED

20: optional double backgroundModelBoost\_DEPRECATED

// -----------------------------------------------------

// Following parameters are used in the FacetsReRanker

// age decay params

14: optional ThriftAgeDecayRankingParams ageDecayParams

// used in the facets reranker

23: optional double maxNormBoost = 5.0

24: optional double globalCountExponent = 3.0

25: optional double simpleCountExponent = 3.0

31: optional ThriftFacetFinalSortOrder finalSortOrder

// Run facets search as if they happen at this specific time (ms since epoch).

32: optional i64 fakeCurrentTimeMs // not really used anywhere, remove?

}(persisted='true')