namespace java com.twitter.recos.user\_tweet\_graph.thriftjava

namespace py gen.twitter.recos.user\_tweet\_graph

#@namespace scala com.twitter.recos.user\_tweet\_graph.thriftscala

#@namespace strato com.twitter.recos.user\_tweet\_graph

namespace rb UserTweetGraph

include "com/twitter/recos/features/tweet.thrift"

include "com/twitter/recos/recos\_common.thrift"

enum TweetType {

Summary = 0

Photo = 1

Player = 2

Promote = 3

Regular = 4

}

enum Algorithm {

Salsa = 0

SubGraphSalsa = 1

}

enum RecommendTweetDisplayLocation {

HomeTimeline = 0

WelcomeFlow = 1

NetworkDigest = 2

BackfillDigest = 3

HttpEndpoint = 4

Poptart = 5

InstantTimeline = 6

Explore = 7

MagicRecs = 8

LoggedOutProfile = 9

LoggedOutPermalink = 10

VideoHome = 11

}

struct RecommendTweetRequest {

1: required i64 requesterId // user id of the requesting user

2: required RecommendTweetDisplayLocation displayLocation // display location from the client

3: required i32 maxResults // number of suggested results to return

4: required list<i64> excludedTweetIds // list of tweet ids to exclude from response

5: required map<i64,double> seeds // seeds used in salsa random walk

6: required i64 tweetRecency // the tweet recency threshold

7: required i32 minInteraction // minimum interaction threshold

8: required list<TweetType> includeTweetTypes // summary, photo, player, promote, other

9: required double resetProbability // reset probability to query node

10: required double queryNodeWeightFraction // the percentage of weights assigned to query node in seeding

11: required i32 numRandomWalks // number of random walks

12: required i32 maxRandomWalkLength // max random walk length

13: required i32 maxSocialProofSize // max social proof size

14: required Algorithm algorithm // algorithm type

15: optional list<recos\_common.SocialProofType> socialProofTypes // the list of social proof types to return

}

struct RecommendedTweet {

1: required i64 tweetId

2: required double score

3: optional list<i64> socialProof // social proof in aggregate

4: optional map<recos\_common.SocialProofType, list<i64>> socialProofPerType // social proofs per engagement type

}

struct RecommendTweetResponse {

1: required list<RecommendedTweet> tweets

}

enum RelatedTweetDisplayLocation {

Permalink = 0

Permalink1 = 1

MobilePermalink = 2

Permalink3 = 3

Permalink4 = 4

RelatedTweets = 5

RelatedTweets1 = 6

RelatedTweets2 = 7

RelatedTweets3 = 8

RelatedTweets4 = 9

LoggedOutProfile = 10

LoggedOutPermalink = 11

}

struct UserTweetFeatureResponse {

1: optional double favAdamicAdarAvg

2: optional double favAdamicAdarMax

3: optional double favLogCosineAvg

4: optional double favLogCosineMax

5: optional double retweetAdamicAdarAvg

6: optional double retweetAdamicAdarMax

7: optional double retweetLogCosineAvg

8: optional double retweetLogCosineMax

}

struct RelatedTweetRequest {

1: required i64 tweetId // original tweet id

2: required RelatedTweetDisplayLocation displayLocation // display location from the client

3: optional string algorithm // additional parameter that the system can interpret

4: optional i64 requesterId // user id of the requesting user

5: optional i32 maxResults // number of suggested results to return

6: optional list<i64> excludeTweetIds // list of tweet ids to exclude from response

7: optional i32 maxNumNeighbors

8: optional i32 minNeighborDegree

9: optional i32 maxNumSamplesPerNeighbor

10: optional i32 minCooccurrence

11: optional i32 minQueryDegree

12: optional double maxLowerMultiplicativeDeviation

13: optional double maxUpperMultiplicativeDeviation

14: optional bool populateTweetFeatures // whether to populate graph features

15: optional i32 minResultDegree

16: optional list<i64> additionalTweetIds

17: optional double minScore

18: optional i32 maxTweetAgeInHours

}

struct TweetBasedRelatedTweetRequest {

1: required i64 tweetId // query tweet id

2: optional i32 maxResults // number of suggested results to return

3: optional list<i64> excludeTweetIds // list of tweet ids to exclude from response

4: optional i32 minQueryDegree // min degree of query tweet

5: optional i32 maxNumSamplesPerNeighbor // max number of sampled users who engaged with the query tweet

6: optional i32 minCooccurrence // min co-occurrence of related tweet candidate

7: optional i32 minResultDegree // min degree of related tweet candidate

8: optional double minScore // min score of related tweet candidate

9: optional i32 maxTweetAgeInHours // max tweet age in hours of related tweet candidate

}

struct ProducerBasedRelatedTweetRequest {

1: required i64 producerId // query producer id

2: optional i32 maxResults // number of suggested results to return

3: optional list<i64> excludeTweetIds // list of tweet ids to exclude from response

4: optional i32 minQueryDegree // min degree of query producer, e.g. number of followers

5: optional i32 maxNumFollowers // max number of sampled users who follow the query producer

6: optional i32 minCooccurrence // min co-occurrence of related tweet candidate

7: optional i32 minResultDegree // min degree of related tweet candidate

8: optional double minScore // min score of related tweet candidate

9: optional i32 maxTweetAgeInHours // max tweet age in hours of related tweet candidate

}

struct ConsumersBasedRelatedTweetRequest {

1: required list<i64> consumerSeedSet // query consumer userId set

2: optional i32 maxResults // number of suggested results to return

3: optional list<i64> excludeTweetIds // list of tweet ids to exclude from response

4: optional i32 minCooccurrence // min co-occurrence of related tweet candidate

5: optional i32 minResultDegree // min degree of related tweet candidate

6: optional double minScore // min score of related tweet candidate

7: optional i32 maxTweetAgeInHours // max tweet age in hours of related tweet candidate

}

struct RelatedTweet {

1: required i64 tweetId

2: required double score

3: optional tweet.GraphFeaturesForTweet relatedTweetGraphFeatures

}

struct RelatedTweetResponse {

1: required list<RelatedTweet> tweets

2: optional tweet.GraphFeaturesForQuery queryTweetGraphFeatures

}

/\*\*

\* The main interface-definition for UserTweetGraph.

\*/

service UserTweetGraph {

RecommendTweetResponse recommendTweets (RecommendTweetRequest request)

recos\_common.GetRecentEdgesResponse getLeftNodeEdges (recos\_common.GetRecentEdgesRequest request)

recos\_common.NodeInfo getRightNode (i64 node)

RelatedTweetResponse relatedTweets (RelatedTweetRequest request)

RelatedTweetResponse tweetBasedRelatedTweets (TweetBasedRelatedTweetRequest request)

RelatedTweetResponse producerBasedRelatedTweets (ProducerBasedRelatedTweetRequest request)

RelatedTweetResponse consumersBasedRelatedTweets (ConsumersBasedRelatedTweetRequest request)

UserTweetFeatureResponse userTweetFeatures (1: required i64 userId, 2: required i64 tweetId)

}