SOCIAL NETWORK BENCHMARK

DATA SCHEMA FEATURES

This document describes the features of the data schema, which are not implicit in the UML class diagram or in the summary table, plus methods for data generation.
These additional features include: value domains, constraints, statistical distributions, and data correlations.

[F1] Relation knows(Person,Person)
This number of friends is generated according to a power-law distribution using a library from university of Montreal
http://www.iro.umontreal.ca/~simardr/ssj/indexe.html

[F2] Relation isLocatedIn(Person,Place)
Set the location Id and also the Z-order of the location by using location dictionary (D1). The distribution follows the population of each country.

[F3] Relation studyAt(Person,University)
Set the university Id for each user according to user's location information. Each location has collection of universities (see the dictionary). The top-10 universities will have much higher probability to be selected than the others (By default, it is 90%).

[F4] Attribute Forum.id
Generate Forum.Id for Wall messages and Status, simply by multiplying the Person.Id with 2. Actually, we do not use the forum for Status Id. We only use the forum for writing post so that the status forum can be removed.

[F5] Relation hasInterest(Person,Tag)
Set the number of interests for each user (i.e., how many singers that he/she likes), and then, generate these interests by using InterestDictionary. There is a correlation between country and the popularity of a singer (Manually collected the popularity by using Google Trends). Then, using exponential distribution for distributing the set of singers for each country. Note that each singer is also associated with a vector of music genres (e.g., POP, ROCK, RAP, HIP-HOP,...) which will be combined to generated the key in the second pass of Map/Reduce job. Specifically, a user may have several singers as his interests. His combined interest score will be the merging of all the vectors from these singers, selecting the highest vector value for each music genre. Then, generate a Z-order value from this merged vector

(See getZValue(MusicGenres musicgenre, int numOfGenres) in ZOrder.java)

[F6] Person's travel frequency (Correlation)
Set whether user is frequent traveller or not. If yes, he will have high probability of changing the location information in his posts.

[F7] Attribute Connection.Browser
Set the browser information (i.e., which web browser that user uses for accessing internet). Randomly generated.

[F8] Relation IPAddress
Generate the IP address for each user. It has the real correlation with the location information (See ipaddrByCountries directory)

[F9] Popular destinations (correlation)
Set the popular places where the user prefers to go to.
This has the correlation with the location information of the user.
Each country has few popular places (2,3 ...) related to F7

[F10] Attribute Company.name
Set the name of the company.
(Location correlated --> have a dictionary)
Dictionary: CompaniesByCountry

[F11] Attribute Person.status (UNUSED)
Set relationship status (e.g., singer, married,...). Have some parameters for the distribution e.g., probSingleStatus. Normally, the probability that a user is singer is the highest.

[F12] Attributes Person.firstName and Person.lastName
These names are location/time/gender correlated.
For each location (country) we only have a set of names. We do not know the gender for that name as well as the period (time) for that set of name. Actually, in dbpedia, the birth years for these names are quite diverse and range for hundred years. So, we did a heuristic for generating the names.

For example, we collect a set of names for a country, sorted by the name's popularity.
N1, N2, N3, N4, N5, N6

We decide that N1, N3, N5 is for Male. N2, N4 and N6 is for female.
From 1980 - 1985, the list of Male name is: N1, N3, N5
From 1985 - 1990, the list of Male name is: N3, N1, N5 (Randomly change the order of few top names comparing the period 1980 - 1985)

From 1980 - 1985, the list of Male name is: N2, N4, N6
From 1980 - 1985, the list of Male name is: N4, N2, N6

As a side note the method used to obtain de random name uses the following distribution:

```
/*
 * If the number of names is smaller than the computed rank
 * uniformly get a name from all names
 * Else, from 0 to (limitRank - 1) will be distributed according to
 * geometric distribution, out of this scope will be distribution
 */
```

[F13] Attribute Person,hasEmail
Generate email (using the dictionary). There are some popular emails such as gmail. Top-5 emails have the popularity scores. Others will be randomly distributed.

[F14] Relation hasTag(Post,Tag)
Tags for a post are selected from user's tag. (May be, with some small probability, the post tag can be randomly selected). Number of tags for a post is <1, number of user's tags>
Content of a post is a fraction from the abstract of the tags