1 Chatous Matching

It is worth going into some more detail about how matching happens on *Chatous*. Every 4 seconds, a batch process is run whereby all of the users who are currently waiting in the queue are matched up. We match users up by first imagining users as nodes in a graph of size n (typically, $n = 100$), and drawing edges between every pair of nodes. The weights on these edges are then used to optimally choose a set of edges such that every user is matched up, and the sum of the edges is maximized. The algorithm we have used is the Blossom Algorithm which, given a general graph $G = (V, E)$, finds a matching such that each vertex in V is present in at most one edge, and the summing of the edges is maximized. This algorithm runs in roughly $O(n^3)$, and we've used the C++ library LEMON implementation of it.

The key part of this matching algorithm is the assignment of weights to edges between every pair of users. At a basic level, we've mandated a few constraints that prevent edges from actually being drawn between every pair of users - these are listed below:

1. No edges drawn between users aged < 18 and aged ≥ 18
2. No edges drawn between users aged < 18 and within 500 miles of each other
3. No edges drawn between users whose profiles contain profane words and users whose profiles are clean
4. No edges drawn between users who are chatting on mobile and who are on desktop

Further, for every weight w we calculate and assign to an edge, we actually normalize each edge that we pass into the Blossom Algorithm to ensure max-cardinality matching, to

$$1 - \frac{2 - 2w}{n}$$

where w is between $[0, 1]$. Thus, each edge weight is between $[1 - \frac{2}{n}, 1]$. This means no node will go unassigned when possible (so we will always prefer to have a matching such that there exists an edge between every pair of nodes rather than a matching where there are unmatched nodes).

We have developed a robust experimental system that allows us to test different algorithms live on the site, that assign weights to edges and measure the global effects. So for instance, if we have x different weighting algorithms we'd like to test, every 4 seconds, with $\frac{1}{x}$ probability, one of the algorithms will be chosen, and the statistics of those users' interactions are kept track of and marked as belonging to that mode.

Our baseline algorithm is following the constraints listed above, and drawing edges between all other pairs of users, assigning a weight of 1 to everyone (i.e. random selection of edges between valid pairs of users). We found this algorithm to produce an average length of conversation of 2.

In the dataset you are working with, we included a few additional heuristics in the matching that goes beyond the baseline:

1.1 Age preferential matching

We define age preferential matching as follows:

$$\text{Age diff } (a, b) = e^{-\frac{|a-b|}{12}}$$

where a and b are the ages of the two users, and the denominator of 12 is used primarily to prevent floating point errors in calculation. Here, we are penalizing matching the further apart their ages. In comparison to the baseline, assigning edge weights in this manner, we found this algorithm to have a 15% improvement (from 2 to 2.3).

1.2 Gender preferential matching

Gender preferential matching is our hypothesis that people of opposite genders prefer to chat to each other. Here, we defined the weight to be:

$$\text{Gender} = \begin{cases} e^{-\frac{3.5}{12}}, & \text{if } a = b \\ 1, & \text{if } a \neq b \end{cases}$$

The numerator in this expression was found through experimental iteration, which basically means that relative to the age difference, we would prefer to pair up users of opposite gender even if they are up to 3 years apart in age. To be clear, the way we calculate weights for a multi feature model such as this is to multiply the weights obtained by each individual parameter together. In comparison to the baseline + age model, this algorithm had a further 15% improvement (2.3 to 2.65).