Algorithms behind LinkedUp's Matching Pipeline

Sam Gu

Feb 2025

1 Methodology

1.1 Per-User Vectors and Scores

1. **Personality Vector**

   - For each user i, define a personality vector $\mathbf{p}_i \in \mathbb{R}^d$.
   - **Example:** we utilize $d = 4$ for MBTI-like dimensions (E/I, S/N, T/F, J/P). Each coordinate $\mathbf{p}_i[k]$ is continuous in $[-1, +1]$ or $[0, 1]$.
   - *Paired with our semantic analysis model, we aim to match users by computing a "synergy measure" in personality space. Specifically, for each dimension $k \in \{1, \ldots, d\}$, we map raw MBTI answers to a continuous scale and define (simplified version of our model):*

$$\text{synergy}_k(\mathbf{p}_i, \mathbf{p}_j) \;=\; \begin{cases} 1 - |\, p_i[k] - p_j[k] \,|, & \text{if focusing on similarity,} \\ |\, p_i[k] - p_j[k] \,|, & \text{if focusing on complementary diversity.} \end{cases}$$

   We can then aggregate across all dimensions (e.g. sum or average) to capture how well two users align or complement each other on personality traits.

2. **Interest Vector**

   - For each user i, define an interest vector $\mathbf{r}_i \in \mathbb{R}^m$.
   - We utilize continuous embeddings, derived from interest tags selected by the user on their user profile, corresponding with our semantic classification algorithms. These embeddings can also be updated based on more communication with people with diverse interests, fostering diverse bond-making.

3. **User Score**

   - A scalar $s_i \in \mathbb{R}$, capturing trust and engagement, useful for moderation and ranking. Typically normalized to $[0, 1]$ or $[0, 100]$.

4. **User Preferences**

   - A small vector $\mathbf{w}_i \in \mathbb{R}^q$, indicating how user i wants to weight personality, interests, or other factors (e.g., a "70% personality, 30% interest overlap" configuration).

1.2 Pairwise Feature Vector

When matching user i with user j, we derive a *pairwise feature vector* $\mathbf{x}_{ij} \in \mathbb{R}^M$. A simplified version is:

$$\mathbf{x}_{ij} = \Big[\underbrace{\text{cosine}(\mathbf{p}_i, \mathbf{p}_j)}_{\text{personality sim}}, \quad \underbrace{\|\mathbf{p}_i - \mathbf{p}_j\|}_{\text{personality distance}}, \underbrace{\text{cosine}(\mathbf{r}_i, \mathbf{r}_j)}_{\text{interest sim}}, \underbrace{\frac{s_i + s_j}{2}}_{\text{score avg}}, \underbrace{|s_i - s_j|}_{\text{score diff}}, \underbrace{(\mathbf{w}_i, \mathbf{w}_j)}_{\text{pref. weights}}, \ldots \Big].$$

1.3 Matching Function / ML Pipeline

We learn or define a function:

$$M : \mathbb{R}^M \;\to\; \mathbb{R},$$

such that $M(\mathbf{x}_{ij})$ outputs a **match quality score**. Higher $M(\mathbf{x}_{ij})$ implies a better predicted outcome (i.e., more successful or satisfying conversation).

1. **Adaptive Feedback**

   Over repeated calls, each user i generates feedback signals $\{y_{ij}(t)\}$ against different partners j at time t. We incorporate these signals via a training set $\{(\mathbf{x}_{ij}, y_{ij})\}$, where $y_{ij} = 1$ if the call was good (or extended), and 0 if it ended poorly. The model is then updated by minimizing a suitable loss function, e.g. cross-entropy:

   $$\mathcal{L}(\theta) \;=\; -\sum\nolimits_{(i,j)\in\text{Data}} \Big[y_{ij} \,\log M_\theta(\mathbf{x}_{ij}) \;+\; (1 - y_{ij}) \,\log\big(1 - M_\theta(\mathbf{x}_{ij})\big) \Big].$$

   This iterative process yields more self-consistent and diverse predictions over time.

2. **Simple Weighted Formula** (hand-tuned):

   $$M(\mathbf{x}_{ij}) = \alpha \,\text{cosine}(\mathbf{p}_i, \mathbf{p}_j) \;+\; \beta \,\text{cosine}(\mathbf{r}_i, \mathbf{r}_j) \;+\; \gamma \,\frac{s_i + s_j}{2} \;-\; \delta \,\|\mathbf{p}_i - \mathbf{p}_j\| + \cdots$$

   where $\alpha, \beta, \gamma, \delta$ are constants or can be dynamically adjusted by user preference vectors $\mathbf{w}_i$, $\mathbf{w}_j$.

3. **Trained Model** (we utilized xgBoost as our PTM):

   $$\hat{y}_{ij} \;=\; M_\theta(\mathbf{x}_{ij}),$$

   where θ are learned parameters. We train M_θ on historical pairs labeled as good/bad conversation outcomes, refining the function M to better capture personality alignment, interest synergy, and user preference signals.

At *inference time*, for a user i seeking a partner, the system computes $\mathbf{x}_{ij}$ for each candidate j in the queue, evaluates $M(\mathbf{x}_{ij})$, and then ranks or filters the candidates accordingly. Over the long run, as feedback accumulates, the model parameters θ and the personality/interest embeddings are updated, producing a more ***self-consistent, accurate, and adaptive*** matching experience.