Suggesting Friends Using the Implicit Social Graph

Yusuf Simonson
The Social Graph
Gmail’s Social Graph
The Problem

- Users do not explicitly maintain group lists
  - Membership changes dynamically
  - Too time consuming and tedious
The Implicit Social Graph

- Billions of vertices – one for each email address
- A group is a unique combination of one or more contacts with whom a user has interacted with in an email thread
The Implicit Social Graph

- Group membership represented through edges
- An active user has on average 350 groups
- Groups have a mean size of 6
- Edges have direction and weight
Interaction Weight

\[ \text{IR} \leftarrow \omega_{\text{out}} \sum_{i \in I_{\text{out}}} \left( \frac{1}{2} \right) \frac{t_{\text{now}} - t(i)}{\lambda} + \sum_{i \in I_{\text{in}}} \left( \frac{1}{2} \right) \frac{t_{\text{now}} - t(i)}{\lambda} \]

- \( I_{\text{out}} \) = set of outgoing interactions
- \( I_{\text{in}} \) = set of incoming interactions
- \( t_{\text{now}} \) = current time
- \( t(i) \) = timestamp of interaction \( i \)
- \( \lambda \) = half-life of interaction weights
Use Cases

• “Don’t forget Bob”
• “Got the wrong Bob?”
Restrictions

• Observability only of a user’s *egocentric network*

• Message contents are not included
Core Routine

function EXPANDSEED(u, S):
  input: u, the user
          S, the seed
  returns: F, the friend suggestions

1. G ← GETGROUPS(u)
2. F ← ∅
3. for each group g ∈ G:
4.     for each contact c ∈ g, c ∉ S:
5.         if c ∉ F:
6.             F[c] ← 0
7.     F[c] ← UPDATESCORE(c, S, g)

• \( S = \) set of contacts that make up the group to be expanded
• \( F = \) map of friend suggestions → confidence score
UpdateScore Implementation #1

- Sums IR scores of all groups with overlap to the seed
- Does not consider the degree of similarity of the seed group to candidate groups
- Biased toward groups with high IR, users in many groups

function INTERSECTINGGROUPSCORE(c, S, g):
  input: c, a single contact
  S, the seed being expanded
  g, a single contact group
  returns: g's contribution to c's score

1. if \( g \cap S \neq \emptyset \):
2.   return IR(g)
3. else:
4.   return 0
UpdateScore Implementation #2

- Scores weighted by similarity of the seed group to the candidate group

```latex
\begin{enumerate}
  \item return $IR(g) \times k|g \cap S|$
\end{enumerate}
```
UpdateScore Implementation #3

- Counts the number of groups a contact belongs to that have any overlap with the seed
- Does not use IR
- Biased toward users in many groups

```plaintext
function INTERSECTINGGROUPCOUNT(c, S, g):
  input: c, a single contact
          S, the seed being expanded
          g, a single contact group
  returns: g’s contribution to c’s score

1. if g ∩ S ≠ ∅:
2.   return 1
3. else:
4.   return 0
```
UpdateScore Implementation #4

- Sum of IR scores for all the groups the candidate user belongs to
- Biased toward frequently contacted users

```python
function TopContactScore(c, S, g):
    input: c, a single contact
          S, the seed being expanded
          g, a single contact group
    returns: an updated rank for the contact c

1. return IR(g)
```
Evaluation

- Randomly sampled real email traffic
- Removed traffic from suspected bots and inactive users
- Sample a few contacts from each group for the seed
- Measure ability of algorithms to account for the rest of the group
Results
Results
Results
“Don’t Forget Bob”

- Straightforward implementation of Friend Suggest algorithm
“Got the wrong Bob?”

- Iterate through each recipient
- Find similarly named recipients
- If their score $> \text{current recipient’s score}$, notify the user

```plaintext
function WRONGBOB(u, L):
    input: $u$, the user
    $L$, a list of the recipients of an email
    returns: a pair $\{c, s\}$ where
        $c$ is a contact $\in L$
        $s$ is a suggested contact to replace $c$

    1. $\text{score}_{\text{max}} \leftarrow 0$
    2. $\text{wrongRecipient} \leftarrow \text{null}$
    3. $\text{suggestedContact} \leftarrow \text{null}$
    4. for each contact $c_i \in L$:
        5. $\text{seed} \leftarrow L \setminus c_i$
        6. $\text{results} \leftarrow \text{EXPANDSEED}(u, \text{seed})$
        7. if $c_i \in \text{results}$:
            8. continue
        9. for each contact $c_j \in \text{results}$:
            10. if $\text{ISSIMILAR}(c_i, c_j)$ and $\text{score}(c_j) > \text{score}_{\text{max}}$:
                11. $\text{score}_{\text{max}} \leftarrow \text{score}(c_j)$
                12. $\text{wrongRecipient} \leftarrow c_i$
                13. $\text{suggestedContact} \leftarrow c_j$
            14. return $\{\text{wrongRecipient}, \text{suggestedContact}\}$
```
“Got the wrong Bob?”
Potential Applications

• Photo, document sharing sites
• IM communication
• Online calendar invitations
• Comments on blog posts
• Text messaging, phone activity
Future Work

• Study relative importance of different interaction types to determine social relationships

• Use of Friend Suggest to identify trusted users
  – Recommendations
  – Content sharing
Q&A