Suggesting Friends Using the Implicit Social Graph

SIGKDD 2010

Maayan Roth, Assaf Ben-David, David Deutscher, Guy Flysher,
Ilan Horn, Ari Leichtberg, Naty Leiser, Yossi Matias, Ron Merom

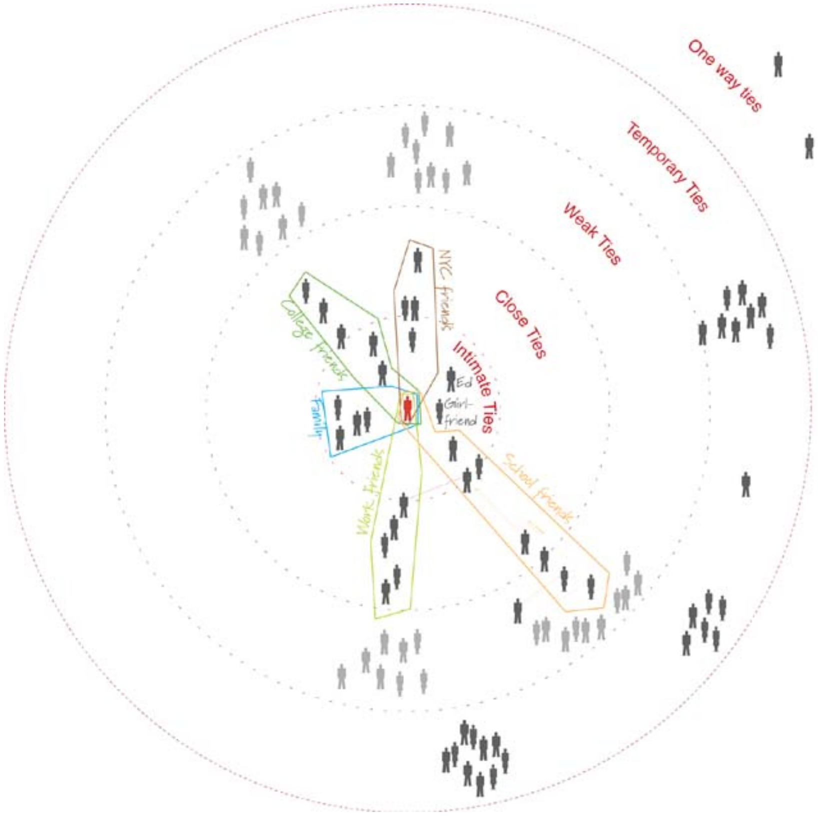
Israel R&D Center, Google

Motivation

- People have 3-5 non-intersecting groups in their social networks

- Users interact with groups
  - Send emails, share photos, plan events

- Tools exist to create persistent contact groups... **but** no one uses them
  - Users say creating groups is “tedious”, “time-consuming”
  - Groups change over time



Credit: Paul Adams, Google UX Research

Automatic group clustering

•**Automatically detect users' contact groups** – help users interact with groups, without forcing them to manually categorize their contacts

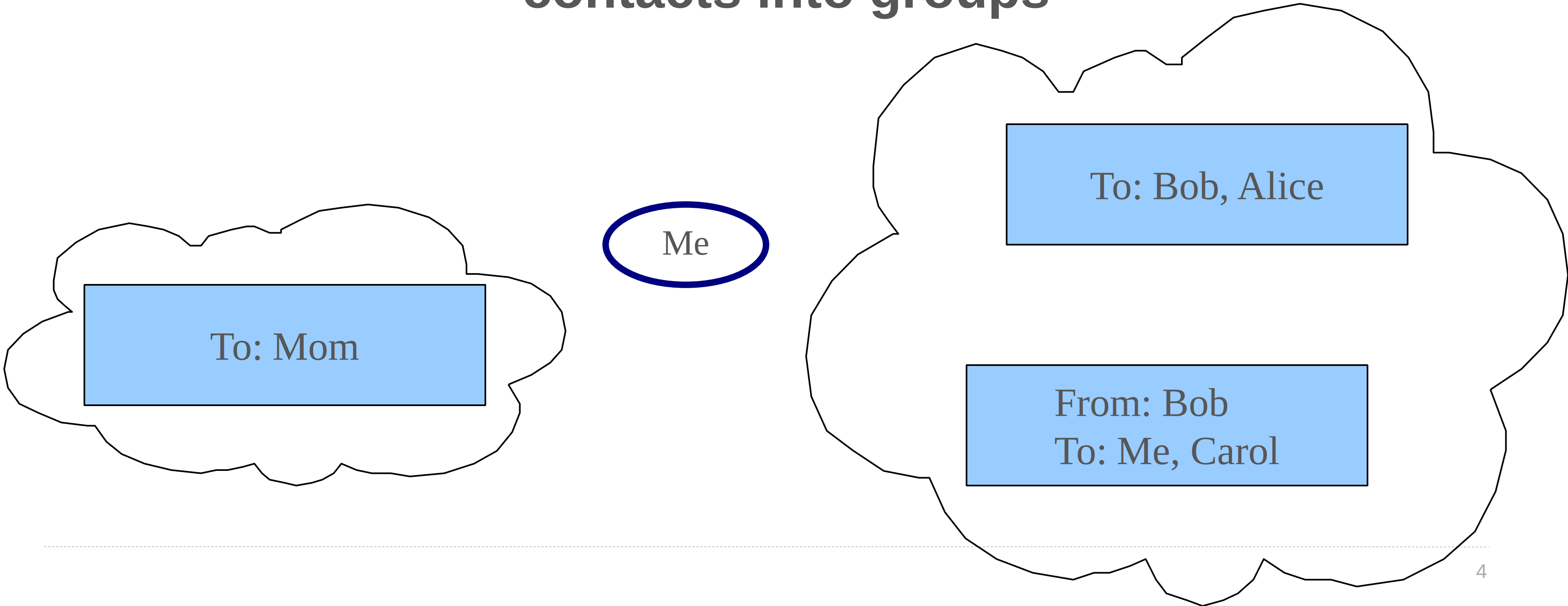
•Previous approaches

- Graph clustering on global social graph – users are likely to belong to the same group if they have many friends in common
  - Kuhn and Wirz “Cluestr: Mobile Social Networking for Enhanced Group Communication” (GROUP '09)
- Content analysis – users are likely to belong to a group if their interactions contain the same keywords
  - Pal and McCallum “CC Prediction with Graphical Models” (CEAS '06)
  - Carvalho and Cohen “Preventing Information Leaks in Email” (SDM '07)

Identifying groups from communication patterns

- Email is a group communication medium
  - Recipient lists are carefully curated
  - Contacts from different groups are unlikely to appear as recipients in the same email

Users' interactions *implicitly* cluster their contacts into groups

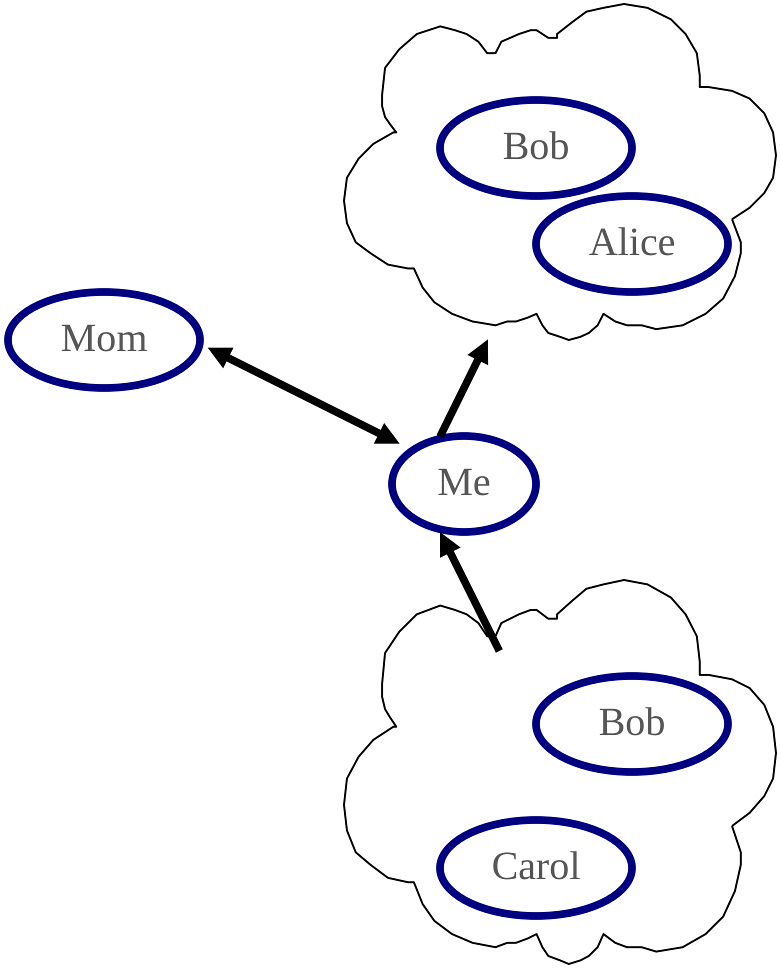


Implicit social graph

- Formed by users' interactions with their contacts
  - Extracted from email headers

- Weighted, directed hypergraph
  - Each email forms a hyper-edge or ***implicit group***

- Egocentric
  - For each user, analyze the subgraph that contains only his direct interactions
  - Addresses privacy concerns



Edge weights

•A user may interact with hundreds of groups, thousands of contacts

- Identify the contacts and groups that are the most important to the user

•Group importance depends on:

- Frequency – More interactions means higher importance
- Recency – Recent interactions more important than those in the past
- Direction – Interactions initiated by the user indicate higher importance

•Summarize all the interactions a user had with a contact or group into a single score, called ***Interactions Rank***

- Weighted count of interactions, where each interaction's weight depends on its timestamp and direction

Edge weights

- Given a set of email interactions between a user and a group *g*:

$$I_g = \{I_{outgoing}, I_{incoming}\}$$

- Define Interactions Rank (IR):

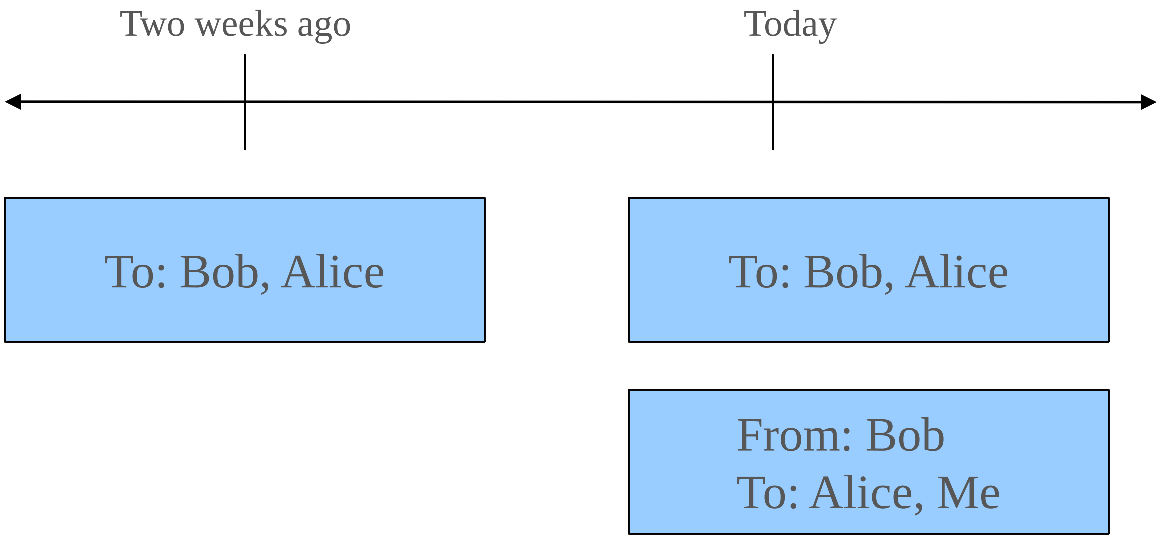
$$\mathcal{IR}(g) \leftarrow \omega_{out} \sum_{i \in I_{out}} \left(\frac{1}{2}\right)^{\frac{t_{now} - t(i)}{\lambda}} + \sum_{i \in I_{in}} \left(\frac{1}{2}\right)^{\frac{t_{now} - t(i)}{\lambda}}$$

- *t(i)* – timestamp of interaction *i*

- *λ* – halflife, determines speed at which an interaction's importance decays

- ω_{out} – weight that determines relative importance of outgoing interaction vs. incoming interaction

Edge Weights



Two weeks ago

Today

To: Bob, Alice

To: Bob, Alice

From: Bob
To: Alice, Me

- If $\lambda = 2$ weeks, $\omega_{out} = 4$:

- Interaction from now has weight 1, then interaction from 2 weeks ago has weight of 0.5
- Outgoing interaction has 4 times the weight of incoming interation from the same time

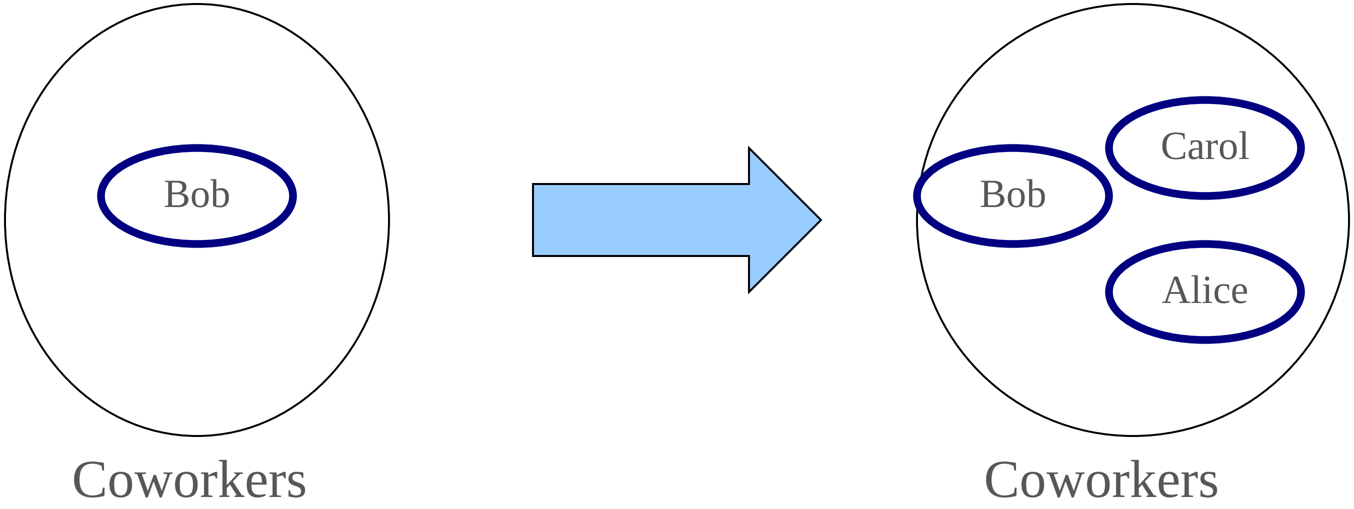
$$IR(Alice, Bob) = \omega_{out}(1 + 0.5) + (1) = 7$$

Finding related contacts

- Given each user's egocentric network, identify groups of related contacts

- Labeling a whole group is time-consuming and tedious... **but** if the user labels a few contacts, maybe we can do the rest

- ***Seed group*** – A small set of contacts, identified by the user as belonging to a semantically meaningful group



ExpandSeed algorithm

- Given a user *u* and a seed group *S*
  - *S* is a small set of contacts identified by *u* as belonging to the same group
  - Return *F*, a set of friend suggestions for contacts that are related to those in *S*
- *G*(*u*) is the set of all groups in *u*'s egocentric network, with their interactions ranks
- For each group *g* in *G*:
  - Iterate over each contact *c* in the group *g*
  - Skip contacts that are already in *S*
  - Compute a suggestion score for the contact *c*, given *g*'s similarity to the seed group *S*, and the interactions rank of the implicit group *g*
- *F*(*c*) stores the sum of *c*'s suggestion scores over all the implicit groups that *c* belongs to
- Return the contacts with the highest scores in *F*

Computing suggestion scores

- For each contact c that belongs to a group g, we know:
  - How similar is the group *g* to the seed group *S*?
  - What is the Interactions Rank of *g*?

- Measure group similarity by looking at the intersection of the group members
  - *g* is similar to *S* if they have contacts in common
  - Similarity increases with intersection size

- We want to know:
  - What is the impact of Interactions Rank on friend suggestion quality?
  - How much does similarity matter?
  - Maybe the user's most important contacts are always good suggestions

What information improves suggestion quality?

- Top Contact
- Considers Interactions Rank, does not consider group similarity

$$w(c) \leftarrow \sum_{g \in \mathcal{G}} \mathcal{IR}(g) \ \mid \ c \in g$$

- Intersecting Group Count

- Considers group similarity, does not consider Interactions Rank

$$w(c) \leftarrow \sum_{g \in \mathcal{G}} 1 \ \mid \ c \in g, |g \cap s| > 0$$

What information improves suggestion quality?

•Intersecting Group Score

- Takes into account both group similarity and Interactions Rank

$$w(c) \leftarrow \sum_{g \in \mathcal{G}} \mathcal{IR}(g) \;\;|\;\; c \in g, |g \cap s| > 0$$

•Weighted Intersecting Group Score

- Takes into account amount of group similarity, and Interactions Rank

$$w(c) \leftarrow \sum_{g \in \mathcal{G}} \mathcal{IR}(g) \times |g \cap s| \;\;|\;\; c \in g$$

Evaluation methodology

All analysis is done in aggregate. We never look at content. We never look at one person's data.

•Build egocentric networks for all users up to time *t*

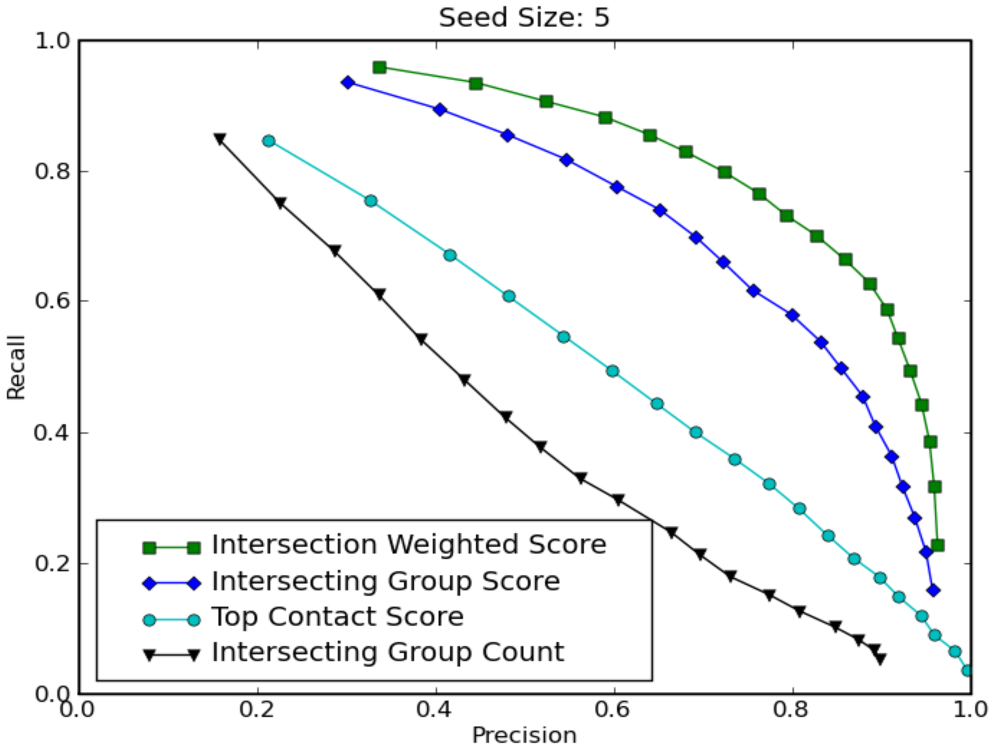
•Randomly sample 100000 new messages with 3-25 recipients, after time *t+1*

- Generate a set of 10000 message headers from emails likely to have been sent by a real person

•For each email:

- Treat the first *k* recipients as the seed set
- Given the seed and the sender's egocentric network, use ExpandSeed to generate seed expansions
- Measure correctness by ability of algorithm to generate remaining recipients

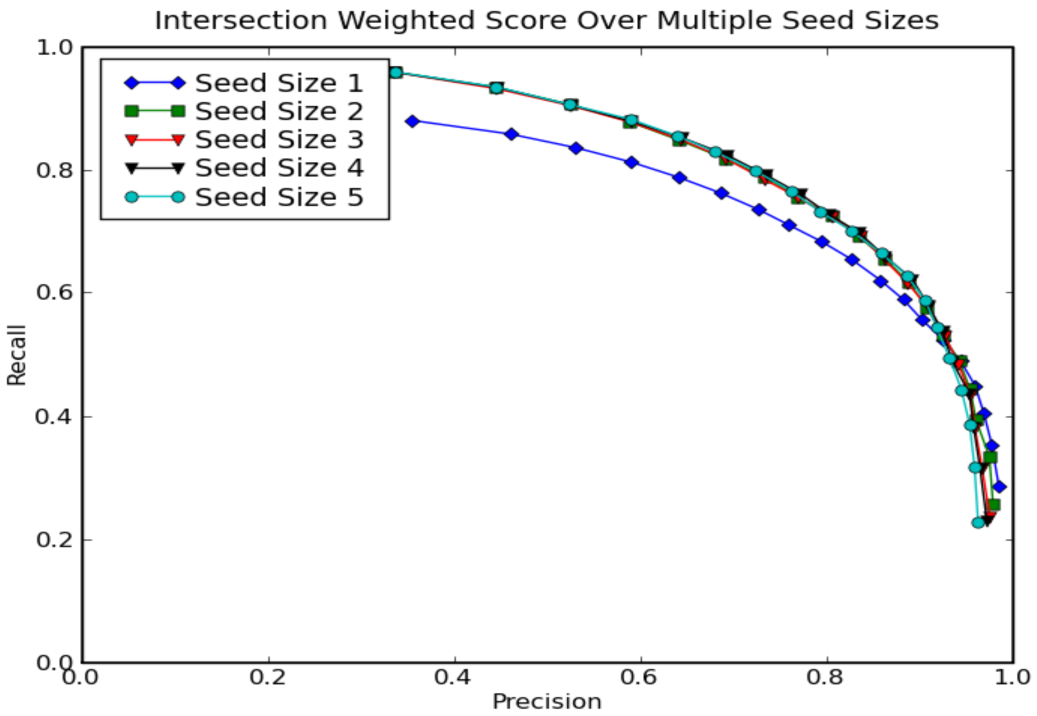
Results



- Taking into account both group weight (Interactions Rank) and group similarity to seed leads to significantly better performance

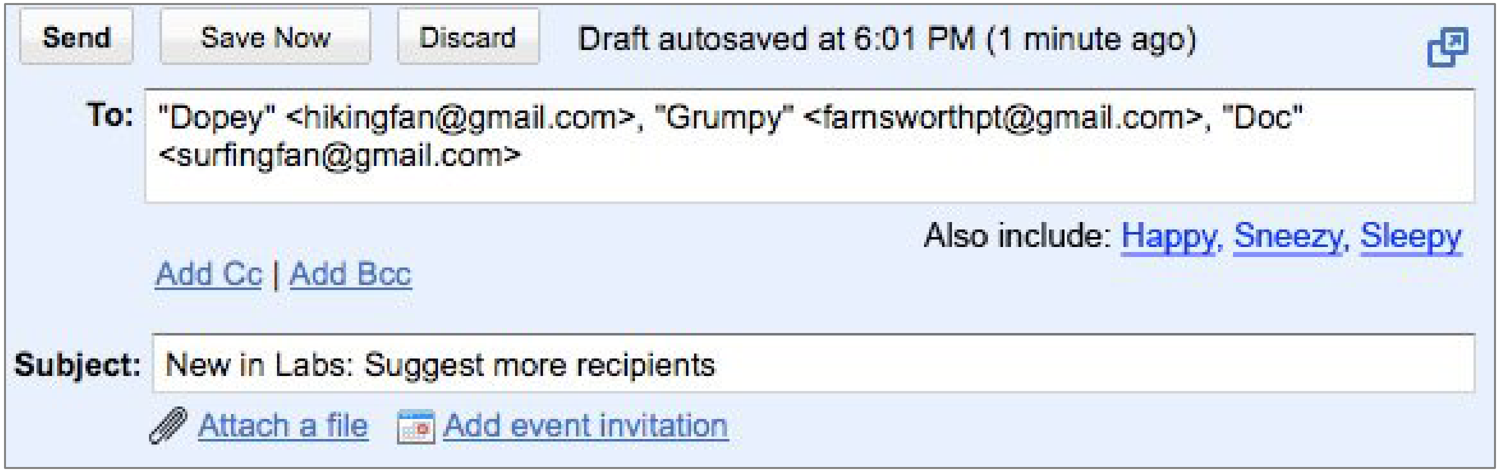
- Best performance also takes into account degree of similarity

Seed size



- For best-performing suggestion score metric, impact of seed size is negligible
- Indicates that groups are well-separated
- Open question for future research

Application - “Don't Forget Bob!”



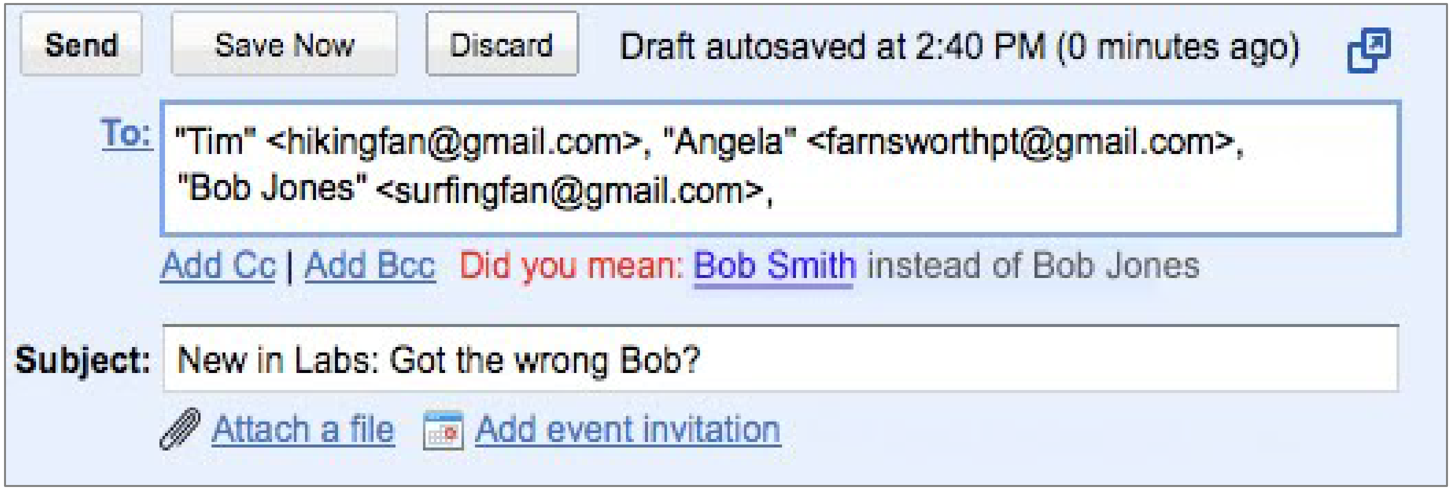
Draft autosaved at 6:01 PM (1 minute ago)

To: "Dopey" <hikingfan@gmail.com>, "Grumpy" <farnsworthpt@gmail.com>, "Doc" <surfingfan@gmail.com>

Also include: Happy, Sneezy, Sleepy

Subject: New in Labs: Suggest more recipients

Application - “Got the wrong Bob?”



Draft autosaved at 2:40 PM (0 minutes ago)

To: "Tim" <hikingfan@gmail.com>, "Angela" <farnsworthpt@gmail.com>, "Bob Jones" <surfingfan@gmail.com>,

Did you mean: Bob Smith instead of Bob Jones

Subject: New in Labs: Got the wrong Bob?

Wrong Bob algorithm

- Given a user *u* and *L*, the list of recipients of an email
  - Return a pair of contacts $\{c_i, c_j\}$ such that:
    - c_i is a contact in *L* that is likely to be a mistake
    - c_j is a suggested replacement for c_i
- For each contact in c_i in *L*:
  - Create a seed group *s* by removing c_i from *L*
  - Generate suggested replacements by running ExpandSeed(*u*, *s*)
  - If c_i is in the suggestion set, it is unlikely to be a mistake
  - Else, compare c_i to the contacts in the suggestion set
    - If there is a contact c_j that is similar to c_i, and has a high suggestion score, keep the pair $\{c_i, c_j\}$ as a candidate
- Return the pair $\{c_i, c_j\}$ for which c_j has the highest suggestion score

Got the Wrong Bob

- Contacts are considered similar if they share a prefix
- Similarity indicates that one of the contacts may have been an autocomplete mistake

- For example, if I send an email to Bob, Alan, and Carol
  - Wrong Bob will suggest “Did you mean to send to Alice instead of Alan?”
  - Alice is a good replacement for Alan because:
    - Alice is a good expansion for the group {Bob, Carol}
    - Alice and Alan share a prefix

- Overall positive user feedback, lots of adoption

Conclusions

•The *implicit social graph* is the graph formed by a user's interactions with his contacts and *implicit groups*

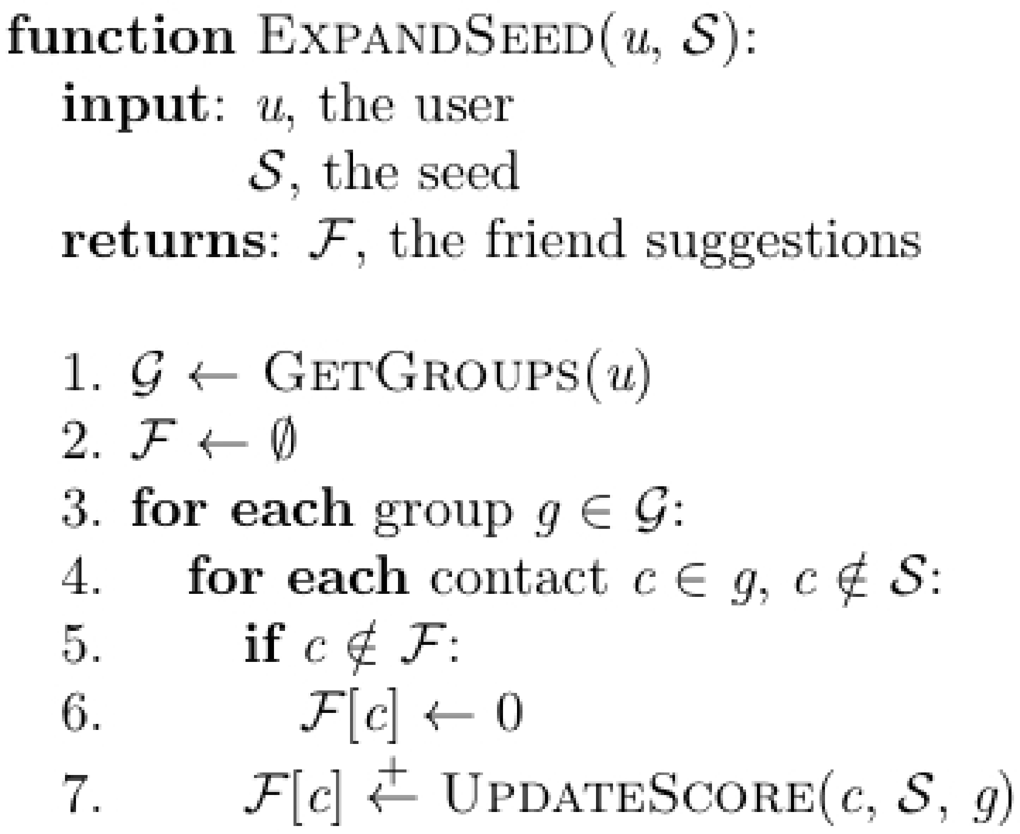
- Weighted, directed, egocentric hypergraph
- Weights based on recency, frequency, and direction of interactions

•Expand a user-specified seed of contacts by looking at similarity to implicit group in user's egocentric network

•Seed-expansion algorithm forms the basis for two successful Gmail Labs, “Don't forget Bob!” and “Got the wrong Bob?”

Extra Slides

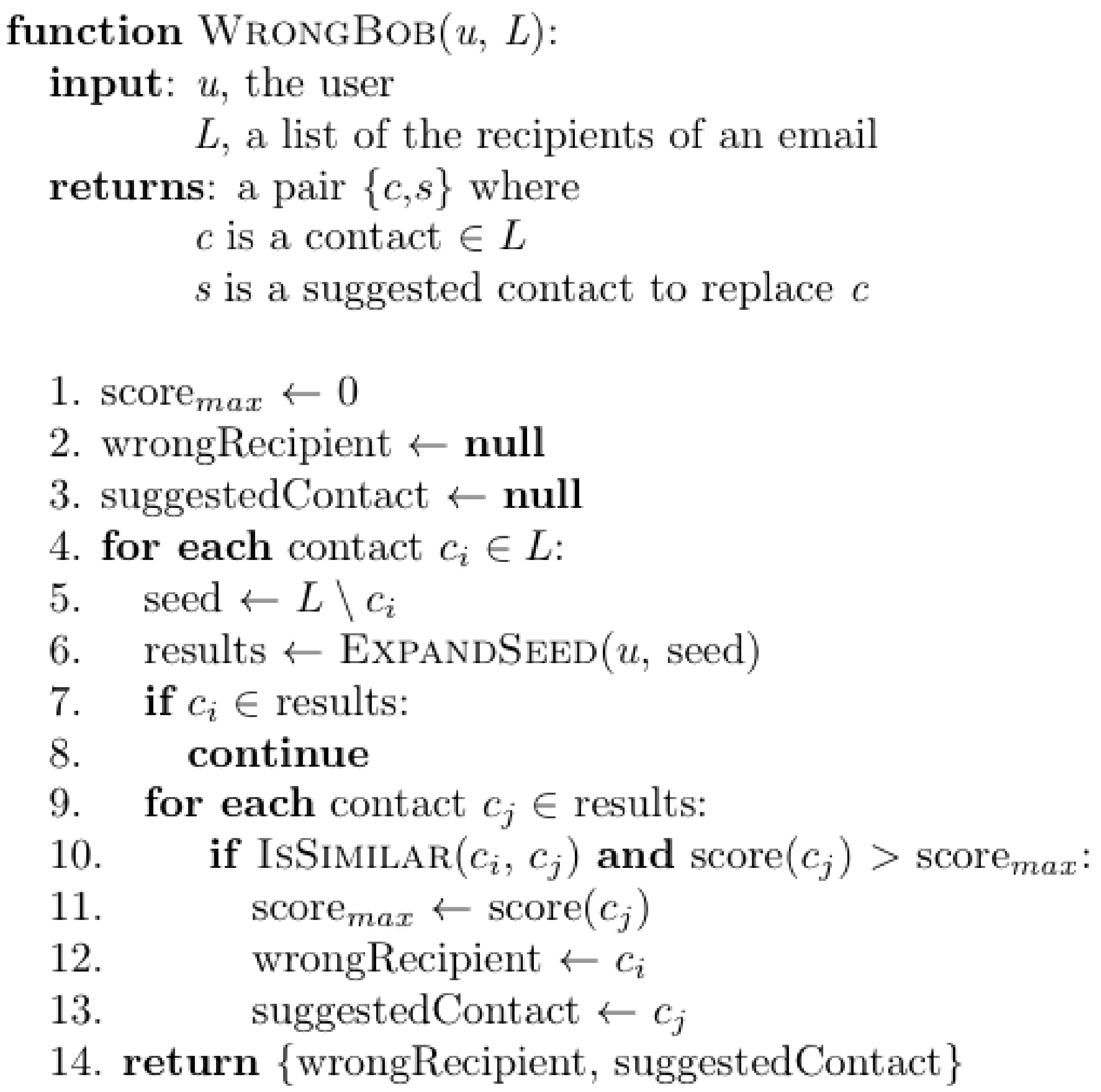
ExpandSeed algorithm



```
**function** ExpandSeed($u$, $\mathcal{S}$):
  **input**: $u$, the user
             $\mathcal{S}$, the seed
  **returns**: $\mathcal{F}$, the friend suggestions

  $\mathcal{G} \leftarrow$ GetGroups($u$)
  $\mathcal{F} \leftarrow \emptyset$
  **for each** group $g \in \mathcal{G}$:
      **for each** contact $c \in g$, $c \notin \mathcal{S}$:
          **if** $c \notin \mathcal{F}$:
              $\mathcal{F}[c] \leftarrow 0$
          $\mathcal{F}[c] \overset{+}{\leftarrow}$ UpdateScore($c$, $\mathcal{S}$, $g$)
```

Application - “Got the wrong Bob?”



```
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 $c$

  $\text{score}_{max} \leftarrow 0$
  wrongRecipient $\leftarrow$ null
  suggestedContact $\leftarrow$ null
  for each contact $c_i \in L$:
      seed $\leftarrow L \setminus c_i$
      results $\leftarrow$ EXPANDSEED($u$, seed)
      if $c_i \in$ results:
          continue
      for each contact $c_j \in$ results:
          if ISSIMILAR($c_i$, $c_j$) and $\text{score}(c_j) > \text{score}_{max}$:
              $\text{score}_{max} \leftarrow \text{score}(c_j)$
              wrongRecipient $\leftarrow c_i$
              suggestedContact $\leftarrow c_j$
  return {wrongRecipient, suggestedContact}
```