Suggesting (More) Friends Using the Implicit Social Graph[*]

[*]

Maayan Roth
mroth@google.com

Tzvika Barenholz
tzvikab@google.com

Assaf Ben-David
abenda@google.com

David Deutscher
dudo@google.com

Guy Flysher
guyfl@google.com

Avinatan Hassidim
avinatan@google.com

Ilan Horn
ilan@google.com

Ari Leichtberg
aril@google.com

Naty Leiser
naty@google.com

Yossi Matias
yossi@google.com

Ron Merom
ronme@google.com

Google, Inc.
Israel R&D Center

ABSTRACT

Although users of online communication tools rarely categorize their contacts into groups such as ”family”, ”co-workers”, or ”jogging buddies”, they nonetheless implicitly cluster contacts, by virtue of their interactions with them, forming *implicit groups*. In this paper, we describe the *implicit social graph* which is formed by users' interactions with contacts and groups of contacts, and which is distinct from explicit social graphs in which users explicitly add other individuals as their ”friends”. We introduce an interaction-based metric for estimating a user's affinity to his contacts and groups. We then describe a novel friend suggestion algorithm that uses a user's implicit social graph to generate a friend group, given a small seed set of contacts which the user has already labeled as friends. We show experimental results that demonstrate the importance of both implicit group relationships and interaction-based affinity ranking in suggesting friends. Finally, we discuss two applications of the Friend Suggest algorithm that have been released as Gmail features.

Categories and Subject Descriptors

H.5.3 [**Information Systems**]: Information Interfaces and Presentation—*Group and Organization Interfaces*;
I.5.3 [**Computing Methodologies**]: Pattern Recognition—*Clustering*

General Terms

Algorithms, Human Factors

Keywords

Implicit social graph, tie strength, contact group clustering.

1. INTRODUCTION

One benefit of many online communication channels over offline methods is that they enable communication among groups of people, rather than restricting communication to be peer-to-peer. Email is just one format that supports group conversations, but there are many others, such as photo- and link-sharing, and collaborative document editing. In fact, group communication is so prevalent that our analysis of the Google Mail email network shows that over 10% of emails are sent to more than one recipient, and over 4% of emails are sent to 5 or more recipients. Within enterprise domains, group communication is even more critical. An analysis of the email network of Google employees showed that over 40% of emails are sent to more than one recipient, and nearly 10% are sent to 5 or more recipients.

As opposed to broadcast-style media, such as blogs[1] and micro-blogging platforms like Twitter[2], the information communicated by an individual to a limited group is generally carefully targeted, and may be private. The recipient lists for small-group communications such as emails are selectively constructed by the message senders. We have observed that users tend to communicate repeatedly with the same groups of contacts. This observation has prompted many online communication platforms to provide their users with tools for creating and saving groups of contacts. Some examples are the Google Mail Contact Manager[3], or custom friends lists on Facebook[4].

Despite the prevalence of group communication, users do

[*] This is an updated version of [16]

[1] e.g. http://www.blogger.com, http://www.wordpress.com

[2] http://www.twitter.com

[3] http://mail.google.com/support/bin/answer.py?
hl=en&answer=30970

[4] http://www.facebook.com/help/#/help.php?page=768

not often take the time to create and maintain custom contact groups. One survey of mobile phone users in Europe showed that only 16% of users have created custom contact groups on their mobile phones [12]. In our user studies, users explain that group-creation is time consuming and tedious. Additionally, groups change dynamically, with new individuals being added to multi-party communication threads and others being removed. Static, custom-created groups can quickly become stale, and lose their utility.

In this paper, we present a friend-suggestion algorithm that assists users in the creation of custom contact groups, either implicit or explicit. This algorithm is based on analysis of the *implicit social graph*, which is the social network that is defined by interactions between users and their contacts and groups of contacts. We differentiate the implicit social graph from explicit social graphs that are formed by users explicitly adding other individuals as ”Friends”. The implicit social graph is a weighted graph, where edge weights are determined by the frequency, recency, and direction of interactions between users and their contacts and groups. Our measure of tie strength differs from previous work in that we consider group interactions, as well as peer-to-peer.

We use the implicit social graph to identify clusters of contacts who form groups that are meaningful and useful to each user. Unlike some previous research on contact clustering, we do not consider the content of interactions. Additionally, because the email network that we have studied is private, we do not consider any friend-of-friend ties, either when computing edge weights for the graph, or when computing contact clusters.

Given a user's social network with weighted edges and an initial seed of a few contacts, our friend-suggest algorithm builds a custom contact group that accurately expands the seed. We evaluate the efficacy of our algorithm by comparing to baseline approaches via precision-recall measurements. We show two applications of this algorithm, implemented as Gmail features, called ”Don't forget Bob!” and ”Got the wrong Bob?” Although our discussion centers around an email network, the network analysis that we have done is applicable to any implicit social graph that is formed by interactions between users and their contacts.

This paper is an extended and revised version of a paper that appeared in KDD 2010. This paper provides additional experimental results about the performance of our seed expansion algorithm. We also provide preliminary empirical results about the performance of the ”Don't Forget Bob!” and ”Got the Wrong Bob?” Gmail features, along with additional discussion.

2. CHARACTERISTICS OF THE EMAIL IMPLICIT SOCIAL GRAPH

The Google Mail implicit social graph is composed of billions of distinct nodes, where each node is an email address. Edges are formed by the sending and receiving of email messages. For the purpose of our work, we consider a message sent from a user to a group of several contacts as forming a single edge, thereby constructing a directed hypergraph. We call the hypergraph composed of all of the edges leading into or out of a single user node that user's *egocentric network.* We call each hyperedge an *implicit group*, even though it may consist of a single contact. On average, a typical 7-day active user has 350 implicit groups in his egocentric network, with groups containing an average of 6 contacts. Note that this does not imply that the average user has thousands of distinct contacts. Rather, each implicit group is a unique combination of one or more contacts with whom the user has interacted.

Edges in the implicit social graph have both direction and weight. The direction of an edge is determined by whether it was formed by an outgoing email sent by the user, or an incoming email received by the user. There may be both outgoing and incoming edges joining a user and an implicit group, if the user has both sent and received email from the group. We consider a user to have received mail from a group by joining the sender of the mail and the other co-recipients into an implicit group. Thus, if a contact c_1 sent mail to the user u and contacts c_2 and c_3, this is represented in u's egocentric network as an incoming edge from the group $\{c_1, c_2, c_3\}$ to u.

The weight of an edge is determined by the recency and frequency of email interactions between the user and the group. In Section 3.1, we propose one metric for computing edge weight, which we call *Interactions Rank.* We claim that edge weight is an important indicator of the strength of the relationship between the user and a particular group. In the remainder of this paper, we use the terms *edge weight*, *group weight*, and *group importance* interchangeably.

In our work, we draw a sharp distinction between each user's egocentric network and the global or *sociocentric* network that is formed by combining the networks of all users. Although other researchers have found value in clustering contact groups by looking at friend-of-friend edges (e.g. [9]), we restrict our algorithm to look only at a single user's egocentric network during friend suggestion. By showing users suggestions based only on their local data, we are able to protect user privacy and avoid exposing connections between the user's contacts that might not otherwise have been known to him.

The social graph studied in this paper is constructed using the metadata (i.e. timestamp, sender, and recipients) of outgoing and incoming messages set or received via Google Mail; message content is not included or examined. For the purposes of this research, we used a random sample of the metadata from thousands of interactions, and data was looked at exclusively in aggregate. The experimental results in Section 4 were gathered with the same privacy protections that are used in all Google software development[5] to ensure that developers do not intentionally or unintentionally access contact information about specific users without their explicit consent.

3. FRIEND SUGGEST

Our algorithm is inspired by the observation that, although users are reluctant to expend the effort to create explicit contact groups, they nonetheless implicitly cluster their contacts into groups via their interactions with them. For ex-

[5] http://mail.google.com/mail/help/privacy.html

ample, while a user may have multiple, possibly overlapping, subgroups of coworkers with whom he exchanges emails, he is unlikely to include his family members in those interactions. The Friend Suggest algorithm, described in this section, detects the presence of implicit clustering in a user's egocentric network by observing groups of contacts who are frequently present as co-recipients in the same email threads. The input to Friend Suggest is a *seed*, which is a small set of one or more contacts that belong to a particular group. This seed could be labeled by the user selecting a few contacts, e.g., as an initial list in the "To:" field of an email. Given this seed, Friend Suggest finds other contacts in the user's egocentric network who are related to the seed, meaning that they are present in the same implicit clusters. Friend Suggest also returns a score for each suggested contact, indicating the goodness of its fit to the existing seed.

The algorithm described in this section is applicable to the problem of group clustering in any interaction-based social graph. For clarity and convenience, we describe it in terms of email interactions.

3.1 Interactions Rank

The first requirement of the Friend Suggest algorithm is an implicit social graph with edges whose weights represent the relationship strength between a user and his implicit groups. We wish to compute edge weights that satisfy the following three criteria:

1. Frequency: Groups with which a user interacts frequently are more important to the user than groups with which he interacts infrequently.
2. Recency: Group importance is dynamic over time.
3. Direction: Interactions that the user initiates are more significant than those he did not initiate.

Regarding recency, we observe that a group with which the user is actively interacting now is more important than one with which the user last interacted a year ago. Overall, recent interactions should contribute more to group importance than interactions in the past. We also note that receiving an email from a contact, a passive interaction, is a weaker signal of closeness than the active interaction of sending an email to that contact. In the most extreme case, we want to be able to rank spammer contacts, from whom the user receives many emails but to whom he sends none, very low in importance.

To satisfy these criteria, we propose Interactions Rank, a metric computed by summing the number of emails exchanged between a user and a particular implicit group, weighting each email interaction as a function of its recency. Interaction weights decay exponentially over time, with the half-life, λ, serving as a tunable parameter. An additional parameter that can be tuned in Interactions Rank is ω_{out}, the relative importance of outgoing versus incoming emails.

Interactions Rank (sometimes abbreviated $\mathcal{IR}$) is computed over a set of email interactions $I = \{I_{out}, I_{in}\}$, according to the following equation:

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

where I_{out} is the set of outgoing interactions between a user and a group, and I_{in} is the set of incoming interactions, t_{now} is the current time, and $t(i)$ is the timestamp of an interaction $i \in I$. Note that according to this equation, an interaction from the current time has a contribution of 1 to a group's Interactions Rank, whereas an interaction from one half-life λ ago contributes $\frac{1}{2}$ and so on.

Interactions Rank is related to the Recency metric proposed by Carvalho and Cohen [5]. However, Interactions Rank calculates the weight of each interaction according to its timestamp, while Recency sorts interactions in chronological order, and weights them on an exponentially decaying scale computed over their ordinal rank. Additionally, Recency does not take into account the direction of each interaction. Ting *et al.* propose an edge-weight metric that considers the role of the interaction participant, but does not take into account the time of the interaction [19].

It should be noted that Interactions Ranks do not easily allow for comparisons across several users. A very active user, who sends and receives many emails per day, will have overall higher Interactions Ranks for his implicit groups than a relatively inactive user. However, within a single user's egocentric network, Interactions Rank allows for a clean ordering of the user's implicit groups by estimated relationship strength. We are actively working on incorporating other signals of importance, such as the percentage of emails received from a contact that the user chooses to read.

3.2 Core Routine

The core routine of the Friend Suggest algorithm, ExpandSeed is shown in Table 1.

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

  1. $\mathcal{G} \leftarrow$ GetGroups($u$)
  2. $\mathcal{F} \leftarrow \emptyset$
  3. for each group $g \in \mathcal{G}$:
  4.    for each contact $c \in g$, $c \notin \mathcal{S}$:
  5.       if $c \notin \mathcal{F}$:
  6.          $\mathcal{F}[c] \leftarrow 0$
  7.       $\mathcal{F}[c] \overset{+}{\leftarrow}$ UpdateScore($c$, $\mathcal{S}$, $g$)
```

Table 1: Core algorithm for suggesting contacts that expand a particular seed, given a user's contact groups.

The ExpandSeed function takes as inputs a user, u, who is the mailbox owner of a single egocentric network in the implicit social graph, and a seed, $\mathcal{S}$, consisting of a set of contacts that make up the group to be expanded. ExpandSeed returns a set of friend suggestions, $\mathcal{F}$, which maps each suggested contact to a score. Each contact's score indicates the algorithm's prediction for how well that given contact expands the seed, relative to the other contacts in u's network. Note that not all contacts from u's network are guaranteed to be returned in $\mathcal{F}$.

Friend suggestions are computed as follows: The user u's egocentric network is extracted from the implicit social graph. The network, $\mathcal{G}$, is represented as a set of contact groups, where each group $g \in \mathcal{G}$ is a set of contacts with whom u has exchanged emails. Each group g has an Interactions Rank, computed as described in Section 3.1, indicating the strength of u's connection to the group g. The goal of ExpandSeed is to find, among all the contacts in $\mathcal{G}$, those whose interactions with u are most similar to u's interactions with the contacts in the seed $\mathcal{S}$.

ExpandSeed iterates over each group g in $\mathcal{G}$, computing a score for each contact c that is a member of g. The algorithm does not suggest contacts that are already members of the seed $\mathcal{S}$. Scores for each contact are computed iteratively via a helper function, UpdateScore, which takes the contact being considered, the contact's score so far, $\mathcal{F}[c]$, the seed $\mathcal{S}$, and the group g. In the following section, we discuss several possible scoring heuristics that were considered for UpdateScore.

3.3 Scoring Functions

UpdateScore is a function template that takes a single contact, c, from a user u's egocentric network and an implicit group g to which c belongs, and returns an incremental score based on the group g's similarity to the seed group, $\mathcal{S}$. The sum of UpdateScore for a contact c over all of the implicit groups to which it belongs is an estimate of c's fitness to expand the seed. Because both the implicit groups making up an egocentric network and the seed group that is the input to Friend Suggest are unordered sets of contacts, they can be compared via standard measures of set similarity [20]. In this work, we look only at set member intersection, leaving more complex metrics for future exploration. We define below several implementations of UpdateScore. In the next section, we evaluate their relative merits.

The most basic instantiation of UpdateScore, shown in Table 2, simply returns a group g's Interactions Rank if the group has a non-empty intersection with the seed set.

```
function IntersectingGroupScore($c$, $\mathcal{S}$, $g$):
  input: $c$, a single contact
         $\mathcal{S}$, the seed being expanded
         $g$, a single contact group
  returns: $g$'s contribution to $c$'s score

  if $g \cap \mathcal{S} \neq \emptyset$:
      return $\mathcal{IR}(g)$
  else:
      return 0
```

Table 2: An implementation of UpdateScore that sums the scores of all of the groups to which that contact belongs, for groups that have a non-empty intersection with the seed.

Intuitively, IntersectingGroupScore finds all the contexts in which the proposed contact c exchanged emails or was a co-recipient with at least one seed group member. However, a larger intersection between the members of the seed group and the members of a given implicit group seems to indicate a higher degree of similarity. Table 3 shows a metric, IntersectionWeightedScore, that takes this into account.

```
function IntersectionWeightedScore($c$, $\mathcal{S}$, $g$):
  input: $c$, a single contact
         $\mathcal{S}$, the seed being expanded
         $g$, a single contact group
  returns: $g$'s contribution to $c$'s score

      return $\mathcal{IR}(g) \times k|g \cap \mathcal{S}|$
```

Table 3: An implementation of UpdateScore that sums the scores of all groups with a non-empty intersection with the seed, weighted by the size of the intersection times some constant k.

We investigate the contribution of group importance to friend suggestion by comparing against a metric, IntersectingGroupCount in Table 4, that simply counts the number of groups a contact c belongs to that have some intersection with the seed $\mathcal{S}$. This metric ignores Interactions Rank entirely, and treats all implicit groups as having equal value to the user.

```
function IntersectingGroupCount($c$, $\mathcal{S}$, $g$):
  input: $c$, a single contact
         $\mathcal{S}$, the seed being expanded
         $g$, a single contact group
  returns: $g$'s contribution to $c$'s score

  if $g \cap \mathcal{S} \neq \emptyset$:
      return 1
  else:
      return 0
```

Table 4: An implementation of UpdateScore that counts the number of groups to which a contact belongs, for groups that have a non-empty intersection with the seed.

Finally, to highlight the importance of using a seed of contacts that characterize a distinct friend group, we compare against an UpdateScore instantiation, shown in Table 5, that ignores the seed and always suggests the top-ranked contacts. Contact ranks are computed by summing the Interactions Ranks of the implicit groups containing each contact.

In each metric, the final friend suggestion scores are normalized with respect to the highest-ranked contact, so that a single threshold can be used across all users, to cut off the list of suggested contacts.

4. EVALUATION

In this section, we evaluate the quality of the Friend Suggest algorithm on real user data. We compare the different scor-

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

    1. return $\mathcal{IR}(g)$
```

Table 5: An implementation of UpdateScore that computes the InteractionsRank of a single contact by summing the scores of all of the groups to which that contact belongs.

ing functions discussed in the previous section, and explore the impact of seed size of friend prediction.

4.1 Methodology

Evaluation is one of the major challenges of developing algorithms that make predictions based on online social network data. Often, researchers build their data sets by surveying a small set of users who are willing to provide the ground truth about their online social relationships [8, 9, 22]. By asking users to categorize their contacts into groups, or rate contacts as ”close to me” or ”not close to me”, researchers can build a labeled data set that serves for both training and testing. However, the nature of this type of survey necessarily limits the number and variety of users who can be included in an experiment. Small sample size and user selection bias can harm the accuracy of the evaluation.

We therefore propose a novel, alternate evaluation methodology. From a stream of real email traffic, we randomly sampled 10000 email interactions with between 3 and 25 recipients. Each recipient list is, in essence, a group of contacts that was implicitly clustered by the user. We can test the accuracy of the Friend Suggest algorithm, and compare the relative success of different scoring functions, by sampling a few recipients from each group, and measuring how well Friend Suggest is able to recreate the remaining recipient list. Our approach is similar to the evaluation methodology used by Pal and McCallum [15], but whereas they removed one recipient from each interaction and verified whether their algorithm could restore him, we begin with small seeds and attempt to generate multiple additional recipients per email.

To generate the 10000 random test interactions, we first sampled 100000 interactions, and then defined rules that aggressively filtered this set to produce a set of email interactions likely to have been generated by active human users. We define an active user as a user with a minimum of 5 implict groups in his social network, who has sent at least one other email in the 7 days prior to the sampled interaction. We attempt to limit our data set to human users by excluding, via regular expression matching, bots and auto-reply addresses such as ”info@domain”, and ”noreply@domain”.

Our experiment tests the ability of our algorithm to use a user's existing social graph to predict his future group interactions. Therefore, when testing our algorithm's ability to predict the remaining recipients in a given email interaction, we use a snapshot of the user's egocentric network based only on interactions that occurred earlier than the sampled interaction.

4.2 Results

Each graph below shows precision-recall curves for the Friend Suggest algorithm using the different scoring functions defined in Section 3.3, with seed groups ranging in size from 1 to 5. For the purposes of our evaluation, we measure precision as the percent of correct suggestions out of the total number of contacts suggested for each seed group, and recall as the percent of correct suggestions out of the total number of email recipients who were not already members of the seed group. A correct suggestion is any contact who was a recipient of the email being evaluated.

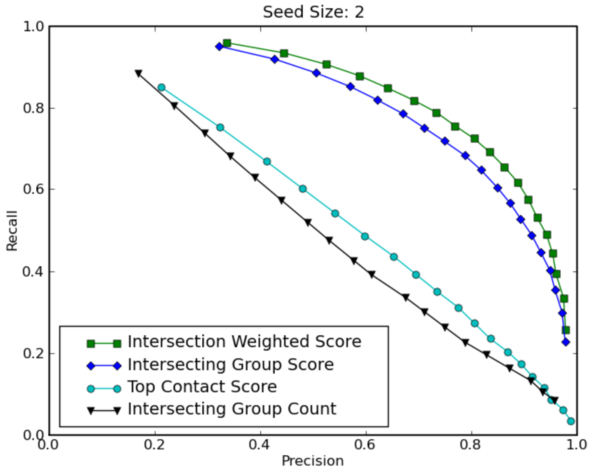


Figure 1: Precision/recall curves for the Friend Suggest algorithm with a seed of 2 contacts, run over the four different scoring functions defined in Section 3.3.

Note that, for all seed sizes, the scoring functions that take into account both group membership and relative group importance, IntersectingGroupScore and IntersectionWeightedScore, significantly out-perform TopContactScore, which ignores the similarity of the seed contacts to the implicit groups and always suggests the top-ranked contacts, and IntersectingGroupCount, which ignores the Interactions Ranks of the groups and simply counts the number of groups in which a contact was a co-recipient with at least one seed contact.

Overall, the scoring function with the best performance is IntersectionWeightedScore. For small seed sizes, its performance is similar to IntersectingGroupScore. However, as the size of the seed contact group increases, IntersectionWeightedScore's performance remains fairly constant, while IntersectingGroupScore's ability to correctly predict email recipients decreases. Because it includes in each contact's score the score of *every* implicit group that contains at least one member of the seed group, IntersectionGroupScore is noisy and prone to false positives. By taking into account the size of the intersection between each implicit group and the seed group, IntersectionWeightedScore is able to discount the impact of spurious implicit

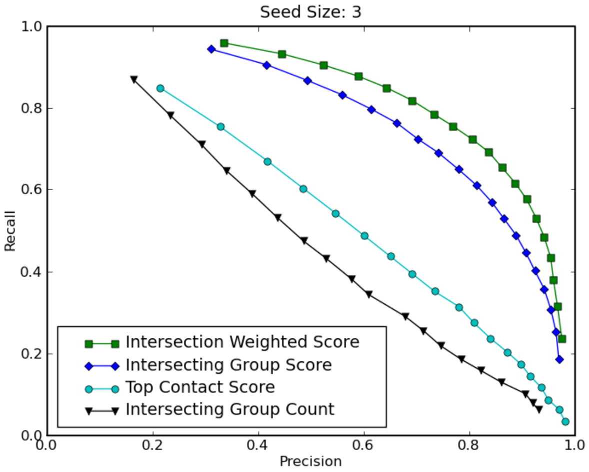


Figure 2: Precision/recall curves for the Friend Suggest algorithm with a seed of 3 contacts, run over the four different scoring functions defined in Section 3.3.

groups that have low similarity to the seed group.

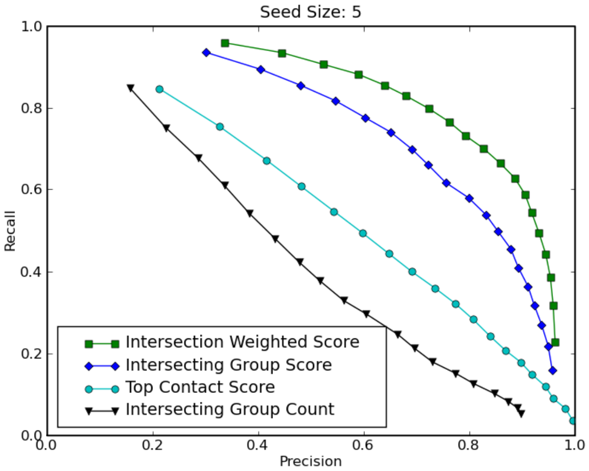


Figure 3: Precision/recall curves for the Friend Suggest algorithm with a seed of 5 contacts, run over the four different scoring functions defined in Section 3.3.

These experimental results demonstrate that the Friend Suggest algorithm, with a correctly chosen scoring function, is able to predict the remaining recipients of an email with high accuracy, given the first few contacts who were added by the user.

4.3 Seed Size Effect

The graphs presented so far show that for every seed size, IntersectionWeightedScore outperforms the other three scores. In this subsection we focus on this score, and analyze the effect of different seed sizes. Figure 4 shows the precision/recall curve for different seed sizes. The size of a seed, with the exception of a seed consisting of a single recipient, has negligible effect on the performance of the algorithm. The high success rates, leads us to believe that given the names of the first two email recipients, one can guess a small set of people, and the email will be sent to a subset of them. For example, if Snow White sends an email to Dopey and Grumpy, it is likely that the rest of the recipient list will only contain dwarfs. However, since she is likely to send messages to different subsets of the dwarves, it is hard to know whether the email be exclusive (for example she may add just one other dwarf), or inclusive, containing most of the dwarves or all of them. Further study is required to address this, perhaps by using better thresholding.

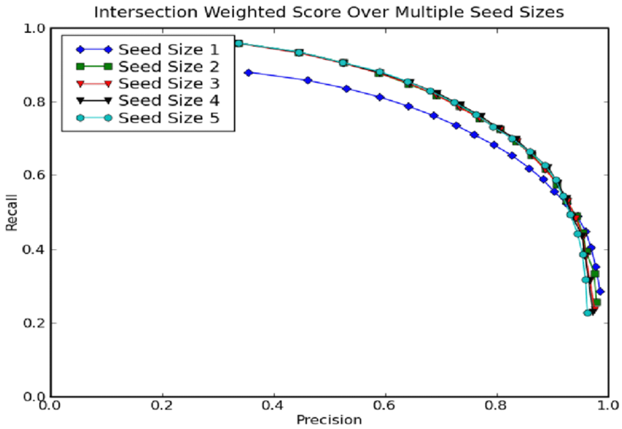


Figure 4: Precision/recall curves for Intersection Weighted Score with seeds of different size.

5. APPLICATIONS

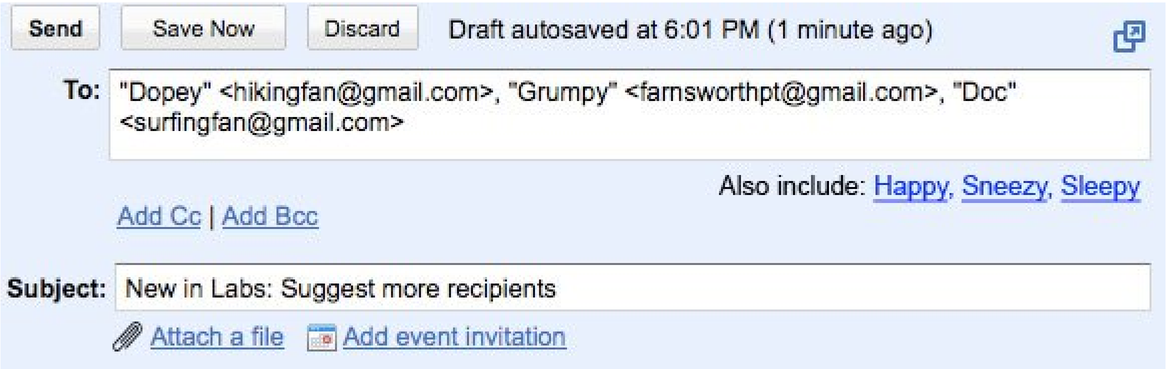
We use the Friend Suggest algorithm in two Gmail features, ”Don't forget Bob!”, and ”Got the wrong Bob?”

5.1 Don't Forget Bob!

”Don't forget Bob” is a straightforward user interface on top of the Friend Suggest algorithm. As seen in Figure 5, ”Don't forget Bob” operates when a user is composing an email message. The feature treats the first contacts added by the user as the seed set, and uses them to generate a set of possible suggested recipients that the user may wish to add to the email. Once the user has added at least two contacts, the application queries the implicit social graph to fetch the user's egocentric network, and uses Friend Suggest to generate up to four contacts who best expand the seed set of existing contacts. These contacts are displayed as clickable links below the ”To:” input field. If the user clicks on a suggestion, or types in another email address, it is added to the list of recipients, and a new set of suggestions is generated.

”Don't forget Bob” has been enabled and used by millions of users, and overall, the user response has been positive. One user posted to the feature's feedback group[6], ”This is incredibly helpful for work/school/family groups without having to create contact groups.”

[6] https://groups.google.com/group/gmail-labs-help-suggest-more-recipients



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

Figure 5: Example screenshot of the ”Don’t forget Bob!” feature in action. Given the user’s initial seed contacts, ”Dopey”, ”Grumpy”, and ”Doc”, the Friend Suggest algorithm suggests additional recipients ”Happy”, ”Sneezy”, and ”Sleepy”.

Coming up with a good quantitative measure for the performance of ”Don’t forget Bob” is non-trivial. One possible metric is to count the number of times users clicked a suggestion, and normalize it by the number of suggestions (or the number of times a suggestion was shown). However, since most users do not click on the suggestions but rather use the keyboard to generate the recipient list, this does not capture the effect of the product. Another alternative is to take the final recipient list, and consider the ratio between the number of recipients who appeared in a suggestion, and the number of suggestions (or the number of times a suggestion was shown). The problem with this type of metric is that over 80% of the email messages are sent to a single recipient, and at every stage it is more likely that the user will hit the send button, than add another name. Thus, to improve the algorithm in this metric, one might be tempted not to show suggestions after the first recipient.

The main metric we use to evaluate the feature counts the number of suggestions accepted, over the number of times at least one suggestion was shown, except in the last step, where the user hits the send button. Suggestions shown at this point do not increase the denominator. Note that when the algorithm shows suggestions it can show up to three suggestions, and still it is only charged once in this benchmark. The results for this metric are very good - the ratio between the number of accepted suggestions and the number of times a suggestion was shown (ignoring the last recipient) is above 0.8. Moreover, this precision comes at a good coverage, and suggestions are shown for more than half the email messages.

5.2 Got the Wrong Bob?

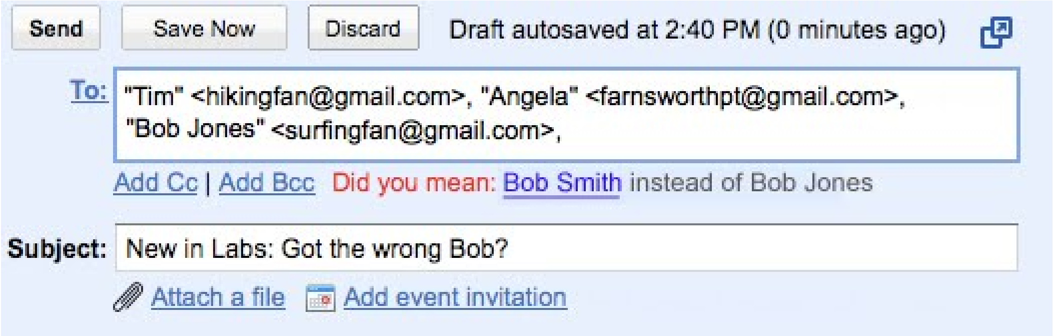
A more complex use of the Friend Suggest algorithm can be found in the ”Got the wrong Bob?” feature, shown in Figure 6. ”Got the wrong Bob” addresses the known problem of email autocompletion errors [4]. While previous approaches have relied heavily on message content, ”Got the wrong Bob” uses the Friend Suggest algorithm to detect the inclusion of contacts in a message who are unlikely to be related to the other recipients.

The WrongBob algorithm, shown in Table 6, works as follows: From the current recipients of an email that have been entered by the user, the algorithm attempts to find a single contact whose removal and replacement with another contact from the user u’s egocentric network would lead to a more coherent recipient list. For each contact c_i in the current recipient list L, WrongBob builds a seed set that includes all of the members of L except c_i (lines 4-5). This seed is expanded via ExpandSeed to generate a set of contacts that are similar to the current members of the seed. If the excluded contact c_i is a member of the suggestion set, it is considered to be related to the other recipients and unlikely to be a mistake (lines 7-8). WrongBob therefore stops searching for a replacement for c_i.

If, however, c_i is not returned as a suggestion from ExpandSeed, it is a potential mistake. WrongBob searches for another contact that could replace c_i. Each contact c_j in the result set returned by ExpandSeed is compared to the error candidate c_i via a helper function IsSimilar. In our implementation, we measured similarity by checking to see if c_j was listed as an autocomplete suggestion at the time that the user entered the contact c_i. If c_i and c_j are similar, and c_j’s score as a member of the seed expansion is higher than the current maximum, c_i and c_j are saved as the current candidate pair (lines 10-13). After examining all contacts in L, the candidate pair with the highest score is returned and displayed to the used as ”Did you mean Contact A instead of Contact B”?

For example, consider the recipient list $L = \{a, b, c\}$. Assume that when removing a to create the seed list $\{b, c\}$, ExpandSeed generates the suggestion set $\{a, d\}$. In this case, because the excluded contact a is a member of the suggestion set, WrongBob determines that it is not a mistake. Then, when removing b, if the algorithm observes $\{b', d\}$, where b' is similar to b but d is not, the algorithm will consider $\{b, b'\}$ as candidates for replacement. If, after removing c, the algorithm generates another candidate pair $\{c, c'\}$, then it will return the pair with the highest score.

Like ”Don’t forget Bob”, the ”Got the wrong Bob?” feature has been enabled and used by millions of users. Additionally, ”Got the wrong Bob?” has received a great deal of favorable media attention in both popular forums such as the New York Times and Esquire, and more technical forums like



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?

Figure 6: Example screenshot of ”Got the wrong Bob?” In the context of an email to recipients ”Tim” and ”Angela”, the Wrong Bob algorithm detects that the user may have intended to include ”Bob Smith” instead of ”Bob Jones”.

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

Table 6: The WrongBob algorithm which, based on the user’s egocentric network, checks if one of the existing recipients would be a good candidate for replacement with another contact.

TechCrunch [3, 18, 17].

As in the previous feature, measuring the true performance is challenging. While some users do click on many suggestions, others prefer to edit email recipient lists via the input textbox after being notified by the the “Don’t forget Bob” feature that they may have made a mistake. As we do in the “Don’t forget Bob“ feature, we consider both actions as a success for the algorithm. In addition, some users only delete the extra recipient, or only add the proposed recipient. In almost 70% of the time, the users accept both suggestions, deleting the wrong Bob and adding the correct one. In almost 90% of the cases, the user accepts at least one suggestion (usually adding the extra Bob). These metrics reflect high precision, but the coverage is about 1%, so only about 1% of the messages trigger the feature. This may sound like a small coverage, but in fact this is not the case. Above 80% of the messages are sent to only one recipient, and thus should not trigger the feature. Out of the messages which are sent to two or more recipients, almost all the messages have the correct list of recipients and again should not trigger the feature. We therefore believe the overall recall of the feature to be high.

6. RELATED WORK

In this section, we discuss related work in three areas: automatic creation of contact groups, analysis of interactions to predict tie strength, and other explorations of communication networks.

6.1 Clustering

There has been some previous work on automatic clustering of online contacts into groups. Reto *et al.* present Cluestr, a clustering algorithm that groups contacts using known graph clustering algorithms [12]. They build an unweighted social graph and cluster it based on edge density between contacts. For example, if a user u has a set of contacts $\{c_1 \ldots c_n\}$ who are highly connected to each other, then they are likely to form a group that is meaningful to u. However, this sociocentric algorithm can only apply to networks in which u is aware of the connections between his contacts. Recall that in our work, we use only the egocentric network of each user, to avoid exposing to the user the private information of his contacts.

In their work on email networks, Pal and McCallum use message content to cluster email recipients into groups [15]. For each user, they build a model that maps keywords and phrases extracted from email messages to the contents who are likely to receive an email containing those terms. They show how these models can be used to successfully predict the recipients of an unaddressed message, and even more successfully, how the ”CC:” and ”BCC:” recipients can be predicted, given the ”TO:” recipients. Carvalho and Cohen use a similar content-based model to solve the inverse problem of finding group anti-members, or email recipients who were likely added to a message by mistake [4]. Unlike these approaches, our analysis is based only on interactions, and disregards content.

The C-Rank algorithm of Bar-Yossef *et al.* is the most similar to the Friend Suggest algorithm described in this paper, in that it is applied to email egocentric networks formed by creating edges between contacts if they appear as co-recipients in email messages [2]. Edges are assigned weights according to the number of email messages involving each pair of contacts. The graph is then thresholded at a number of different edge weight thresholds, with edges falling below the threshold removed, to created a set of unweighted graphs. C-Rank identifies clusters of contacts within these graphs by finding *vertex separators*, or contacts whose removal from the graph creates disconnected subgraphs. The authors claim that a good cluster is one that exists in several graphs with different threshold levels.

Within enterprise networks, where expectations of privacy are lower than in consumer email networks, researchers have used sociocentric analysis to cluster and classify groups of users. For example, De Choudhury *et al.* use an inferred social network constructed from email interactions to assign roles to the various participants, such as "student", "faculty", or "director" [6]. They use email frequency to filter noisy and potentially spurious edges from the graph.

6.2 Tie-Strength Prediction

Another body of related work has explored the use of interactions as a signal for measuring the strength of social ties. Much of this work has studied the relationship between Facebook users and their contacts. Gilbert and Karahalios use a set of 70 different features to predict the strength of connection between a user and his Facebook friends [8]. These features range from demographic features of the users, such as age and religion, to interaction frequency and recency features such as the number of comments a user has left on his friend's photos and the time elapsed since their last email exchange. They found that the strongest predictor of tie strength between two individuals is a short elapsed time between message exchanges.

In other work on Facebook tie strength prediction, Kahanda and Neville compare the relative predictive value of interactions, which they call *transactional features*, as compared to graph topology or profile attribute features [9]. They found the highest predictive value in *network-trans-actional features*, which extend transactional features to include friend-of-friend links. For example, "Person X posted on Person Y's wall" is a network-transactional feature about the relationship between Person X and some other Person Z, if Y and Z are friends. Xiang *et al.* also build a predictive model of the strength of ties on Facebook, looking at interactions such as face-tagging in photos [21].

In email networks, researchers have primarily been interested in tie-strength as a useful feature for predicting the emails to which a user is likely to reply [10]. Yoo *et al.* find that including social features along with message content-based features in the vector of classifier input led to a significant reduction in prediction error when learning to identify the emails that a given user will consider important [22].

6.3 Communication Networks

Finally, there has been significant recent interest in exploring and understanding the properties of communication networks. Ting *et al.* propose a general-purpose architecture for extracting communication-based social networks from multiple sources of interaction data [19]. Leskovec *et al.* survey a large number of different communication networks to answer the question of whether they share a common community structure [14]. An in-depth study of one particular explicit network, the social network formed by MSN Instant Messenger users, is performed by Leskovec and Horvitz [13]. They find that users tend to communicate most frequently with users who are demographically similar to themselves. Furthermore, they find that the overall network is robust to the removal of random nodes, in that it does not disturb the overall connectivity of the graph.

Email networks are the most commonly studied implicit social graphs. Dodds *et al.* show that email networks follow a small-world property, with an average shortest path of 4.1 steps between any two nodes [7]. Adamic and Adar further explore strategies that users can employ to exploit this dense connectivity in order to efficiently propagate information through an enterprise email network [1]. Kossinets and Watts study the social network formed by email exchanges between students and faculty of a large university, and find that, although the local connections between individuals evolve over time, the overall network structure remains stable [11].

7. CONCLUSIONS

In this paper, we studied the *implicit social graph*, a social network that is constructed by the interactions between users and their groups. We proposed an interaction-based metric for computing the relative importance of the contacts and groups in a user's egocentric network, that takes into account the recency, frequency, and direction of interactions. We then defined the Friend Suggest algorithm which, given a single user's egocentric network with computed edge weights and a seed set of a few labeled contacts, finds other contacts who are related to the seed contacts, and therefore form a semantically meaningful group. We demonstrated the effectiveness of the Friend Suggest algorithm via a novel experimental methodology. Finally, we showed two applications of the Friend Suggest algorithm, the Gmail features "Don't forget Bob!" and "Got the wrong Bob?", and presented some data on their performance.

Although the experimental results described in this paper were performed by examining email interactions from the Google Mail system, the algorithms and approaches described in this paper apply to any interaction-based social network. Some other interaction types that could form similar implicit networks are photo and document sharing, instant messenger chatting, online calendar meeting invitations, or comments on blog posts. Even offline interactions, such as mobile text messages or telephone calls, form an implicit social graph between individuals and groups. Our future research is intended to study the relative importance of different interaction types in determining the social relationships between individuals. We are also interested in exploring other applications of the Friend Suggest algorithm, such as identifying trusted recommenders for online recommendation systems, or improving content sharing between users in various online contents.

8. REFERENCES

[1] L. Adamic and E. Adar. How to search a social network. *Social Networks*, 27, 2005.

[2] Z. Bar-Yossef, I. Guy, R. Lempel, Y. S. Maarek, and V. Soroka. Cluster ranking with an application to mining mailbox networks. In *Proceeding of the IEEE International Conference on Data Mining (ICDM)*, December 2006.

[3] J. D. Biersdorfer. Tip of the week: Got the wrong Bob? http://www.nytimes.com/2009/10/22/technology /personaltech/22askk-003.html, October 22 2009.

[4] V. R. Carvalho and W. W. Cohen. Preventing information leaks in email. In *Proceedings of the Seventh SIAM International Conference on Data Mining (SDM)*, April 2007.

[5] V. R. Carvalho and W. W. Cohen. Ranking users for intelligent message addressing. In *Proceedings of the 30th European Conference on IR Research (ECIR)*, March 2008.

[6] M. D. Choudhury, W. Mason, J. Hofman, and D. Watts. Inferring relevant social networks from interpersonal communication. In *Proceedings of the 19th International World Wide Web Conference (WWW)*, April 2010.

[7] P. S. Dodds, R. Muhamad, and D. J. Watts. An experimental study of search in global social networks. *Science*, 301, 2003.

[8] E. Gilbert and K. Karahalios. Predicting tie strength with social media. In *Proceedings of Computer Human Interaction (CHI)*, April 2009.

[9] I. Kahanda and J. Neville. Using transactional information to predict link strength in online social networks. In *Proceedings of the Third International Conference on Weblogs and Social Media (ICWSM)*, June 2009.

[10] T. Karagiannis and M. Vojnovic. Behavioral profiles for advanced email features. In *Proceedings of 18th International World Wide Web Conference (WWW)*, April 2009.

[11] G. Kossinets and D. J. Watts. Empirical analysis of an evolving social network. *Science*, 311, 2006.

[12] M. Kuhn and M. Wirz. Cluestr: Mobile social networking for enhanced group communication. In *Proceedings of the International Conference on Supporting Group Work (GROUP)*, May 2009.

[13] J. Leskovec and E. Horvitz. Planetary-scale views on a large instant-messaging network. In *Proceedings of the 17th International World Wide Web Conference (WWW)*, April 2008.

[14] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. In *Proceedings of the 17th International World Wide Web Conference (WWW)*, 2008.

[15] C. Pal and A. McCallum. CC prediction with graphical models. In *Proceedings of the Third Conference on Email and Anti-Spam (CEAS)*, July 2006.

[16] M. Roth and A. Ben-David and D. Deutscher and G. Flysher and I. Horn and A. Leichtberg and N. Leiser and Y. Matias and R. Merom Suggesting Friends Using the Implicit Social Graph 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining KDD 2010.

[17] M. Siegler. Meeting with the Bobs? Gmail will now make sure you contact the right one. http://www.techcrunch.com/2009/10/13/meeting-with-the-bobs-gmail-will-now-make-sure-you-contact-the-right-one/, October 13 2009.

[18] M. Sullivan. Self-correcting e-mail: Because you can drunk-dial at the office, too. http://www.esquire.com/blogs/endorsement/gmail-got-the-wrong-bob-101309, October 13, 2009.

[19] I.-H. Ting, H.-J. Wu, and P.-S. Chang. Analyzing multi-source social data for extracting and mining social networks. In *Proceedings of the International Conference on Computational Science and Engineering*, 2009.

[20] R. E. Tulloss. Assessment of similarity indices for undesirable properties and a new tripartite similarity index based on cost functions. In M. E. Palm and I. H. Chapela, editors, *Mycology in Sustainable Developement: Expanding Concepts, Vanishing Borders*, pages 122–143. Parkway Publishers, 1997.

[21] R. Xiang, J. Neville, and M. Rogati. Modeling relationship strength in online social networks. In *Workshop on Analyzing Networks and Learning with Graphs*, December 2009.

[22] S. Yoo, Y. Yang, F. Lin, and I.-C. Moon. Mining social networks for personalized email prioritization. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, June 2009.