Towards Composable Prediction of Contact Groups

Andrew Ghobrial, Jacob W. Bartel, Prasun Dewan
Department of Computer Science
University of North Carolina
Chapel Hill, NC USA
{andrewwg, bartel, dewan}@cs.unc.edu

***Abstract*—Users' contacts often need to be grouped into equivalence classes for various purposes such as easily sending a message to all members of the group. Several approaches have been recently developed to make such predictions (a) for both ephemeral and persistent groups (b) in both email and social networks systems. However, no research has attempted to compare these approaches or compose them by using ideas of one in another. We have taken a step in this direction. We have developed and compared multiple approaches to predicting persistent contact groups in email. These approaches compose an algorithm that generates friend lists in Facebook from a social graph with different techniques for generating the social graph. One of these techniques is based on a scoring algorithm used by Google to predict ephemeral groups incrementally. To compare the approaches we ran a user study involving 19 participants and used two simple metrics that calculated the average percentage difference between a predicted group and the group of addresses in a future message. The evaluation showed that using the Google score was the best approach though it offered very small improvements over all but one of the simpler methods.**

Keywords-email; prediction; user-groups

I. INTRODUCTION

Users' contacts often need to be grouped into equivalence classes for various purposes such as sending a message to the group; sharing files, posts, or photos with the group; filing messages related to the group; and understanding the social networks to which users belong [2]. Therefore, a variety of systems such as social networks, email, and file and database systems allow users to create these classes.

However, in order to use these classes or groups, users must first incur the cost of identifying and creating these groups. Research has shown that few groups are actually created by users in a variety of systems, such as Facebook [2, 15], email [9, 14], and mobile phones [8]. As past work has observed, there are many reasons users may fail to create these groups. For example, users may not understand how to create groups [11] or they may find the process of creating groups tedious, difficult, or time-consuming [11, 14].

As a result, a variety of research efforts have developed algorithms for recommending both ephemeral and persistent groups of users in a variety of systems [1, 2, 4, 6, 7, 9, 10, 12, 14, 16]. These efforts are exciting because they allow us to show that automatic prediction offers substantial benefits over manual composition of these groups. However, no research has attempted to compare these different prediction techniques or compose them by using ideas of one in another. In this paper, we describe an initial step in this direction.

A large number of compositions are possible even with the relatively small number of approaches addressing this emerging area. Arguably, the two most diverse of these are (a) an approach developed by Bacon and Dewan [2] that uses friend relationships in Facebook to recommend persistent groups in "batch" without using any information about the use to which these groups are to be put, and (b) an approach developed by Roth et al [14] and implemented in Gmail that, given a specific email and a set of known correct recipients, incrementally predicts candidates for the next recipient. We utilized (parts of) these two approaches to make a new kind of prediction: prediction of persistent named contact in email, thereby meeting the composition requirement. To meet the comparison requirement, we composed the Facebook approach with two simpler versions of the Google approach and compared all three compositions to determine the usefulness.

Our work required us to address several new issues:

- How exactly should the Facebook and Gmail approaches be composed?
- What are some simpler but potentially useful versions of the Google approach that could be used for the comparisons?
- How should we determine values of tunable parameters of the composed approaches?
- What criteria should be used to compare the approaches?
- What are the results of the comparison?

In the rest of the paper, we address these issues. In the following section we overview existing group-prediction techniques. Next, we motivate and describe the three compositions. Then we describe how we tuned our parameters and made our comparisons. Finally, we present conclusions and directions for future work.

II. RELATED WORK

There are a variety of approaches that mined and/or recommend groups of users for a variety of systems [1, 2, 4, 6, 7, 9, 10, 12, 14, 16]. Of these approaches, some have focused on groups to be used for roles in role-based access control [16], some have focused on groups in social networks such as Facebook or Google+ [1, 2, 4, 7, 10], and a few have focused on groups in email [6, 14].

Only some of these approaches [1, 2, 4, 7, 9, 10, 16] have recommended groups directly to users so that users may edit and label the groups. Other approaches have mined groups for the understanding and organization of networks [6, 12]. One approach developed by researchers at Google does not pre-create the groups, but rather suggests a user to add to the recipient list one at a time based on groups it automatically identified [14]. This work was even expanded later to use these automatically identified groups to automatically group recommended recipients of a message into a hierarchy [3].

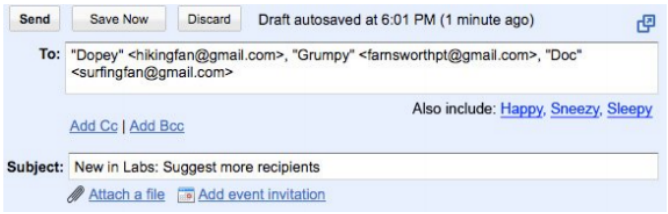

Figure 1. Gmail recipient recommendation interface

Our research focuses on prediction of persistent groups in email. E-mail is one of the most prevalent forms of communication on the Internet. It is estimated that in 2013, around 183 billion e-mail messages were sent per day [13], many of which were addressed to several recipients. Instead of addressing each recipient, many systems such as Gmail, Microsoft Outlook, or Mozilla Thunderbird allow users to address messages to groups of recipients.

However, in order to use these groups, users must first incur the cost of identifying and creating groups. One way to reduce this cost is by automatically recommending the e-mail sender with groups based off the users' past e-mail exchanges. For example, if a user regularly communicates with their parents and grandparents together via email, an algorithm should detect this strong link between the user, their parents, and their grandparents, and thus present them as a group suggestion. This group of parents and grandparents can then be used both to address new email messages in the future and to filter new incoming messages by performing an action such as sorting them into a folder called "Family".

This recommending of groups does not come without costs to users. Once users are presented with a suggested group they must assign a label, such as "family", to the group and possibly make modifications to the membership of the group in the form of adding or deleting certain recipients. For example, the algorithm might only have suggested a user's parents and grandparents, but the user may also want to include his/her siblings. Alternatively, the user may choose to reject the group entirely if he/she feels that it will not be useful in the future.

The best prediction algorithms will produce groups that are least likely to be rejected by the user. Moreover, of the groups that are accepted by the user, ideally the user should have to make the least additions or deletions.

However, determining the best algorithms requires a comparison of alternative approaches. Moreover, to carry this field forward, it is important to understand the similarities between them so that we can differentiate between components that are competing alternatives and those that are complementary and thus can be composed. However, most of the research mentioned above was done independently and more or less contemporaneously, without considering other work with similar goals.

Composition and comparisons both require identification of similarities and differences between these approaches. To our knowledge, all approaches providing automatic group identification have required graphs representing relationships between users or recipients in order to identify groups. In these graphs, nodes represent users or recipients, and nodes have edges between them if the respective users or recipients have a relationship. These graphs are then mined to determine groups.

Thus, past work in automatic group identification can be organized into a two-dimensional design space. One dimension is the method for creating graphs, and the second dimension is the method for mining graphs. Past work has covered a variety of portions of each of these dimensions.

In some cases, forming the required graphs is a relatively simple task. For example, consider the domain of social networks including Facebook and Google+. To form graphs, users of a social network can be represented as nodes, and if two users are explicitly linked (such as by being friends on Facebook or being contained in each other's circles on Google+), an edge is created between them in the graph. However, in systems such as email or file systems, this is made more difficult by the lack of clear relationships between users. Users are merely implicitly linked by being addressed in the same e-mail exchange or have access to the same file. Therefore, past work has used many different approaches for creating graphs. For example, some of these approaches [6, 9, 14] create edges between two users if they are recipients of some of the same email messages, but drop edges if the messages were not sent recently enough or were not included in enough messages together.

Past work has also employed a variety of approaches to predict these groups. It has used gradient ascent to form groups that are most likely to exist given the current graph [10] or identified small candidate groups in the graphs such as individual nodes or maximal cliques, and merged candidates based on the distances between them, such as Jaccard similarity [9], cohesion [7], or required additions and deletions [2, 4].

Despite the wide coverage of this design space of automatic group identification, when restricted to the domain of email, the design space has significantly less coverage. In fact, to our knowledge only MacLean et al. [9] have automatically identified persistent groups that are then presented to the user. They determined candidate groups such as unique recipient sets (or the union of the `TO`, `CC`, and `BCC` fields) from past sent messages. They then removed any candidate groups that were only included in a small number of messages according to some threshold and merged groups that either led to an information leak less than some threshold or had a Jaccard similarity above some threshold.

As we see above, both the creation and mining of the graph is email-specific in that it uses messages addresses to both determine the graph and mine it. Thus, it is not clear how this approach should be composed with other approaches to

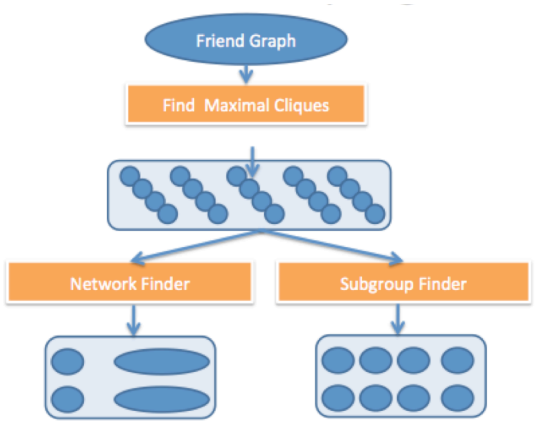

Figure 2. Hybrid Clique Merger approach

recommend persistent groups that are based on friend relationships rather than users addressed in email together.

Based on the discussion above, however, the reverse is possible. It is possible to use email-specific information to create a social graph that is then mined using a recommender for friend lists. This is the composition approach we took, discussed in more depth below.

III. APPROACH

The Facebook mining algorithm we used is called Hybrid Clique Merger [2, 4] and shown in Figure 2. It finds the maximum cliques in the graph where a clique is a set of vertices in which each vertex is connected to every other vertex by an edge. Overlapping cliques are combined to form large groups called networks in the graph. Networks containing more than 50 members are treated as their own subgraph. Subgroups are then found by finding maximal cliques in the subgraph and merging them.

In this algorithm, the more strongly connected a set of vertices is to each other, the more likely that a group containing these vertices will be recommended. For example, if three Facebook users were all friends with each other, a group would be recommended containing all three. If the three friends were also strongly connected to another friend, he/she would also be included in the group.

Since this approach was designed and originally tested in social networks, successful group prediction in email may imply a strong link between groups in email and social networks. Thus, if we are able to make successful group predictions in email with this approach, it leaves open more directions for future work to cross-pollinate concepts involving groups in email and social networks.

However, as mentioned previously, the use of this approach in email is made difficult by the lack of clear relationships between recipients. Recipients are only linked because they are merely addressed in the same e-mail exchange (as compared to the clear user to user friend relationships on Facebook and other social media sites).

Thus, our next step was to determine an effective way to generate a graph that will produce optimal groups. Keeping with our goal of comparing different points in the design space of automatic group detection, we designed, implemented, and studied three different graph generation algorithms, described in the three subsections below.

A. Simple Graph Generation:

This approach works by using every past email message. For each message, a vertex in the graph is created representing a collaborator in the message. A collaborator is either a sender or recipient of the message. An edge is then created between every pair of people listed as collaborators of the message. Thus, an edge represents a connection between two people, and a connection is created for each pair of users involved in a message. For example, consider the following message recipients:

```
From: james@univ.edu
To: alice@cs.univ.edu, zach@cs.univ.edu
```

The vertices `"james@univ.edu"`, `"alice@cs.univ.edu"`, and `"zach@cs.univ.edu"` are created. The following edges are also created:

- `("james@univ.edu" <--> "alice@cs.univ.edu")`
- `("james@univ.edu" <--> "zach@cs.univ.edu")`
- `("alice@cs.univ.edu" <--> "zach@cs.univ.edu")`

A total of $\frac{n\,(n-1)}{2}$ edges are created for a message with n collaborators. Thus the number of edges increases on the order of $O\ (n^2)$. A potential downside of this approach is that it does not take into consideration the age of the message.

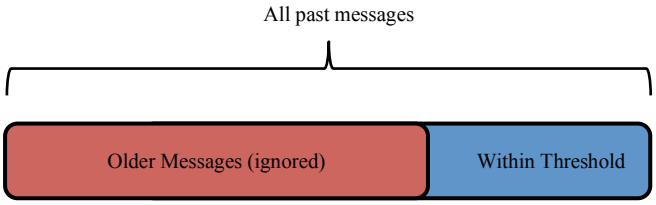

Figure 3. Simple Threshold illustration

B. Simple Threshold Graph Generation:

This graph generation works similarly to the above, but also takes in as parameters a threshold age in milliseconds and a date argument. In determining whether to use a message in the graph, the algorithm checks to see if the date of the message is more than the threshold age before the date passed in as an argument.

This filtering of past messages is represented by Figure 3. The right, blue area represents the size of the threshold age (in milliseconds). Any message in the left red area is a message that is considered too old to be useful in generating groups and is thus completely ignored in generating the graph and groups.

C. Google Scoring Algorithm:

This next algorithm meets our goal of composing work from both Facebook and email group prediction. It is based on Gmail's recipient recommendation algorithm [14]. In this algorithm, the next set of recipients is recommended based on an automatically constructed graph. In this graph, nodes representing recipients or groups of recipients have weighted

edges in between them based a score called Interactions Rank (IR). This score is computed based on a formula that factors in the half-life of a message and whether it was sent or received in order to create a score for each edge. The formula, shown below, then sums the weights of these messages.

$$IR = \omega_{out} \sum_{m \in M_{out}} \left(\frac{1}{2}\right)^{\frac{t_{now} - t(m)}{\lambda}} + \sum_{m \in M_{in}} \left(\frac{1}{2}\right)^{\frac{t_{now} - t(m)}{\lambda}}$$

In this formula, M_{out} is the set of past sent messages, M_{in} is the sent of past received messages, $t(m)$ is the time of message m, t_{now} is the current time, λ is a half life weight, and ω_{out} is a sent message weight.

We compose the Google approach with the (Hybrid) Clique Merger by using this Google core, not to compute the weighted graph used for recipient prediction but to compute an un-weighted graph to be mined by the Clique merger. Thus, our (version of the) Google Scoring Algorithm uses the half-life of a message and whether it was sent or received in order to create a score for each edge in the Clique Merger graph. Therefore, unlike the Simple Graph Generation algorithm, it takes into account time, and unlike the Simple Threshold Graph Generation algorithm, it takes into account whether a message is sent or received and treats older messages as less important rather than ignoring them entirely.

This algorithm starts by created a *weighted graph* in which each edge is assigned a score using the above formula. After the algorithm finishes creating the weighted graph, it converts the graph into an un-weighted graph by dropping edges below a certain threshold. The intuition behind this approach is that older edges are assigned a decreasingly smaller edge weight instead of using a hard cut-off as with the Simple Threshold algorithm. For example, consider two collaborators of a message, Alice and Zach, and a half-life of one week. If one message occurred 1ms ago between them, the weight score from this message would be 1. If a message were also sent between Alice and Zach one week ago, then the weight score from this earlier message would be 0.5. The new weight of the edge between Alice and Zach would be the sum of the weights for the two messages, or 1.5.

IV. STUDY

A. Data Collection Framework

We needed data to not only compare these three approaches, but also determine appropriate values for their tunable parameters. As our goal is to predict persistent groups from email histories, we created a framework for collecting email histories without compromising the privacy of the subjects. We created a tool that collected e-mail message data anonymously in the format shown in Figure 4.

In this format, each line represents a single e-mail message. Each message is given a Message Id and Thread Id. Also collected is the From Id, set of Recipient Ids and the received date of the message. After users log in with their e-mail address and password, a preset amount of the most recent e-mail messages is collected in this anonymous format from their accounts. Note that each From Id and Recipient Id represents an actual e-mail address in the header of the message such as james@univ.edu. Recall that the received date is used in determining the importance of older messages relative to more recent ones.

```
Message:1 Thread:1 From:[1] Recipients:[2] Received-Date:Fri Feb 07 07:47:40 EST 2014
Message:2 Thread:2 From:[3] Recipients:[1] Received-Date:Fri Feb 07 05:00:17 EST 2014
Message:3 Thread:3 From:[1] Recipients:[4,5,6] Received-Date:Thu Feb 06 23:46:23 EST 2014
Message:4 Thread:3 From:[4] Recipients:[1,5,6] Received-Date:Tue Feb 04 16:44:59 EST 2014
Message:5 Thread:4 From:[7] Recipients:[8] Received-Date:Thu Feb 06 20:13:17 EST 2014
```

Figure 4. Format of anonymous email message data

We wrote a parser to create a Message object for each line in the file containing the message properties discussed above. Thus, when given the above file as input, a List<Message> object is returned containing all the Message objects. This object contains all the data we require to recommend the set of groups for a particular user.

The email collector accepted both Gmail and Microsoft Outlook/ Live addresses because both are commonly used by undergraduate students, graduate students, faculty, and staff at our institution. It gave users the option of recording private email addresses, in which case the actual Id to e-mail address mappings were collected and stored in a separate, secure file, the format of which is shown in Figure 5.

```
278:james@univ.edu
276:fall_cohort@univ.edu
247:alice@cs.univ.edu
155:doug@alumni.univ.edu
253:bob@univ.edu
125:zach@cs.univ.edu
```

Figure 5. Format for mapping anonymous ids to email addresses

The file allowed us to manually evaluate our own groups by seeing if they make sense logically in lieu of running the testing algorithm described later. This file was also parsed and stored as a Map<Integer, String>. We then created the graph using the Strings of the actual e-mail addresses. If private data was not available, we simple created the graph using the String representation of the integer.

B. Automated Data Collection

Past work on persistent groups that involved users to manually create or edit groups has reported that many subjects were not willing to put in this effort In particular, MacLean et al. [9] indicated that the vast majority of subjects did not manually create their own groups, and Bacon and Dewan [2] report that only about half of the subjects edited the groups that were predicted automatically, though those that did found this effort to be small. Therefore, we decided to run a fully automated study, using objective metrics, that involved no user effort beyond going to a web link, signing an IRB consent form, choosing the options, and entering email id and password. In response to our announcements about the study, we were able to collect data from 19 participants. For each user, up to 400 threads containing a total of up to 2000 messages were collected. These values were picked so that as many e-mail messages for each thread are collected as possible. On average between 500 and 600 messages were collected from each user using these parameters. The users were mostly undergraduate students at our university.

C. Determining Constants and Thresholds

As mentioned earlier, in order to effectively apply the graph creation approaches we previously described, we had to first determine appropriate weights and thresholds for some of the approaches. In the case of the Simple Threshold Graph approach, we needed to determine an effective time threshold. In the case of the Google Scoring Algorithm, we needed to determine an effective weight for sent messages, half-life weight, and threshold for edge scores. These values are not reported in the paper that introduced this algorithm [14].

For the Simple Threshold Graph approach, we considered the thresholds of 1 hour, 1 day, 1 week, 2 week, 1 month, and 2 months in pilot testing. We generated groups from our own e-mail accounts with these thresholds and checked their usefulness. We found that groups generated with a 1 hour, 1 day, or 1 week threshold were unlikely to produce any useful groups. This is reasonable because 1 hour or 1 day is not a long enough time period to be able to generate persistent groups that are useful in the long term. Based on these findings, we chose the thresholds of two weeks, one month, and two months to further test using the data collected from the user study.

The Google Scoring Algorithm required us to pick three constants: a half-life, a sent constant, and an edge weight threshold at which to drop edges. In order to find the best combination of these constants, one could simply perform a brute force search across all possible combinations of all possible values of these constants. However, given that there may be many possible values for each of these constants, such a search is not practical. Moreover, the large number of tests we would need to run predicting groups using all possible constant values would make it likely that our results were good based on chance rather than having found an effective set of constants to predict groups.

Therefore to set each of these three parameters, we performed tests using our own data. In these tests, we only varied one chosen parameter and fixed the other two values. This allowed us to determine the effect of changing the chosen parameter on the edge weight distributions and therefore select an acceptable value. A parameter is not very useful if it has little effect on the distribution. In other words, if it yields edge weights such that a vast majority of weights are close to each other - edges with close weights would be included or excluded together.

We evaluated this effect of different parameter values using a cumulative distribution function (CDF) plot of edge weights. In this plot, possible edge weights are along the X axis and the percentage of edges having a weight less than or equal a given weight are along the Y axis. We then displayed multiple, different colored plots on a single graph, where each plot corresponds to the CDF of edge weights for a chosen parameter value. If the CDF for a specific parameter value is more towards the upper left-hand corner or the lower right-hand corner of the plot, then there is little variation in edge weights, meaning it is likely not possible to determine a good edge-dropping threshold. If it is in the upper left-hand (lower-right hand) corner, most of the edges have a small (large) weight; and the edge weight threshold parameter would not be very good at discriminating among the edges, regardless of its value.

To test half-life constants, we considered half-lives of 1-hour, 1-day, 1-week, 2-weeks, and 1-month. The CDF plot of the edge scores using these half-lives and a sent constant of 1 are shown in Figure 6. As the figure illustrates, both 1-hour and 1-day half-life constants have CDF that are close to the upper left-hand corner of the graph. As mentioned previously, this indicates that there is little variation in edge weight and therefore it may not be possible to specify an effective threshold. Comparatively, the 1-week, 2-weeks, and 1-month half-life constants were more towards the center of the graph, indicating a greater variation on edge weights and a better possibility of choosing an effective threshold. Therefore, we chose the second, more successful, set of values in our group evaluation described later.

To analyze the sent constants, we performed a similar analysis using a fixed half-life of one week and sent constants of 1/16, 1/8, 1/4, 1/2, 1, 2, 4, 8 and 16. A sent constant of 2 means that sent messages edges are given twice the weight. Intuitively, a message that is sent should be given more consideration when generating groups than a message that was received as it defines a group from the point of the sender – the user for who the groups are being predicted - rather than the receiver.

As demonstrated by the CDF plot in Figure 7, there was very little variation in the edge weights across any of the sent constants. This indicates that changing the sent constant has little effect on the edge weights in the graphs we constructed. Based on this limited effect, we then decided to use 1 as the value of this parameter, which means sent messages are equally important as received messages in predicting groups.

Finally, we needed to answer the question, "at what point do we drop edges?" To do so, we fixed the sent constant as 1 and the half-life constant as 1-month, which was one of our successful approaches. The CDF of the edge weights of these constants is shown as the dark blue plot in Figure 6, which is the rightmost distribution in Figure 6

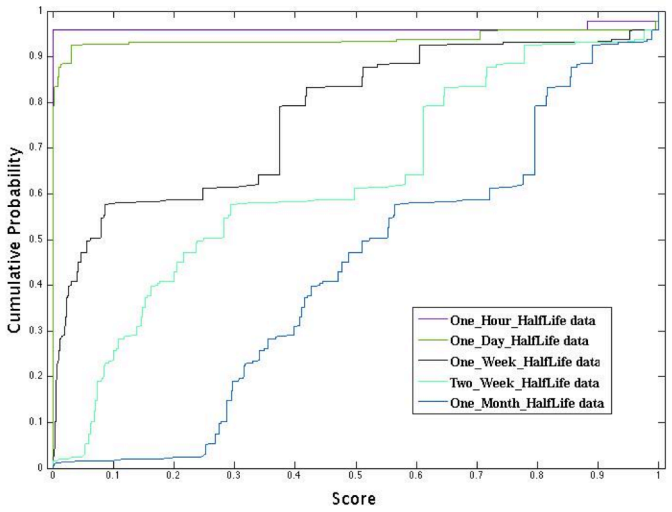

Figure 6. CDF for same sent-constant, various half-lives

As the figure shows, there are a few elbows in the graph, or points at which there is a stark change in the derivative. In particular, the first of these elbows occurs at approximately 0.25. Before this point, the value and derivative is close to 0, indicating there are some, but relatively few, edges below this threshold. Because the number of edges with thresholds higher than 0.25 increases after this point, it is likely that the edge weights below this points are heavily influenced by noise rather than any meaningful signal. Moreover, since the edge weights are likely noise, the edges themselves are likely noise. Since the goal of the threshold is drop superfluous edges, we chose 0.25 as our edge weight threshold to drop these likely noisy edges.

Our next task was to define metrics for comparing the three schemes. Previous research has used two approaches for evaluating how much effort automatic group prediction saves over manual group composition: (a) a subjective evaluation of the effort saved based on task completion time and user interviews [9], and (b) an objective evaluation of the effort saved by asking subjects to morph predicted lists into ideal lists and measuring the number of edits required in this task. As mentioned earlier, both evaluations were heavyweight in that they involved significant effort, which several of the recruited subjects were not willing to put in.

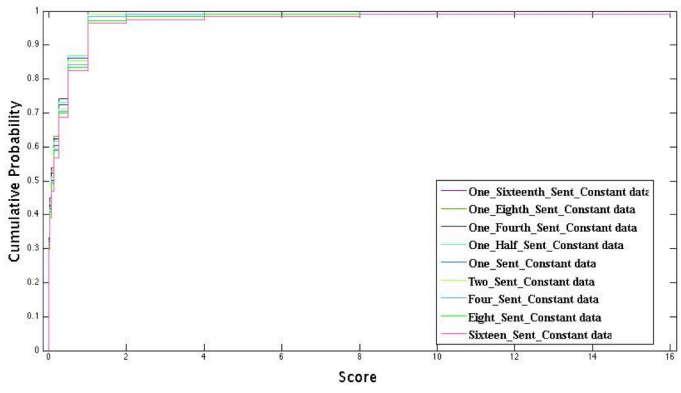

Figure 7. CDF for same half-life, various sent-constants

As we were predicting in email, we essentially had data about ideal lists through the users grouped in email messages. We developed metrics based on these data that attempted to compute the benefit of using predicted groups in future messages, that is, messages generated after the groups were predicted.

D. Training and Testing Sets

Thus, computation of these metrics involves prediction of groups based on certain messages and then determining how well these groups work for messages generated after the prediction. We used the standard approach of dividing each user's messages into training and testing sets.

However, we could not divide the data arbitrarily as the training messages must occur chronologically before the testing messages. In a realistic scenario, users would use our prediction schemes to generate groups based on past messages for use in future messages. If our results are to match reality, our predictions must mirror such a scenario. For this reason, we could not perform k-fold cross validation. With multiple passes, in some passes, some tests messages would occur chronologically before training message. Even if such tests resulted in effective group prediction for the test messages, it is not clear that they would match reality,

Therefore, for each user, we sorted each participant's list of messages in chronological order. Each participant's messages were split into a training and test set. The training set contained the first 80% of messages and the testing set contains the remaining 20% of messages. The algorithm predicted groups using the training set, and we then evaluated the usefulness of the predicted groups when applied to the testing set.

We defined two metrics: a group-centered metric and a message-centered one.

E. Group-Centered Evaluation Metric

This metric attempts to find how useful a predicted group is in *some* future message, that is, a message sent after the group is predicted.

For each group, g, we computed the distance of each message to the group. For our purposes, distance was measured as the number of edits (additions/deletions), required to transform the message collaborators (sender + recipients) to the group. We chose this definition of distance, because it matches Bacon and Dewan's [2] model of how users edit members of recommended groups with additions and deletions.

Based on this distance measure, we found the message, m, with the minimum distance. We divided this minimum distance by the number of collaborators in m to give the cost of using the group in the best message. This computation is described by the following equation, with g being a predicted group, and M be the set of test messages.

$$GroupCenteredMetric(g) = \frac{min_{m \epsilon M} distance(g, m)}{collaborators(m)}$$

Table 1. Graph Generation Constants

Graph Generation Approach	Time Threshold	Half-life	Sent Constant
Simple Graph	-	-	-
Simple Threshold	2 Weeks	-	-
Simple Threshold	1 Month	-	-
Google Score	-	1 Week	1.0
Google Score	-	2 Weeks	1.0
Google Score	-	1 Month	1.0
Google Score	-	2 Months	1.0

For each approach, we computed the mean value of all predicted groups for the approach. The closer the group-centered metric is to 0, the more effort saving the prediction of the group offers in comparison to manually creating the group of collaborators in the message. However, if the metric is greater than one, the group is useless since for all the messages it takes fewer (insert and delete) operations to manually enter the recipients than use the group.

F. Message-Centered Evaluation Metric

The value of the metric above would be high even if a group is used in a single message. Thus, if there are G predicted groups and M future messages, and M >> G, then the groups could conceivable be useful for only G of M messages.

Therefore, in addition to the group-centered evaluation metric, we used a message-centered evaluation metric, which evaluates on average how close the best group is to each future message. In this approach, we are testing the usefulness of the best group for each message (the reverse of section group-centered). For each message, m, we found the distance of each group from the message. Again, distance was the number of edits (additions/deletions), required to transform the message collaborators (sender + recipients) to the group. We then found the group with the minimum closeness. This minimum closeness was then divided by the number of collaborators in m. Thus, this metric gave us the relative cost of using the best group in the message. Again we represent this with an equation, where m is a message and G is the set of predicted groups:

$$MessageCenteredMetric(m) = \frac{min_{g \epsilon G} distance(g, m)}{collaborators(m)}$$

As with the group-centered metric, for a given approach, we computed the mean value for all test messages. This was then used as the message-centered metric for that approach. Again, if this value is higher than 1, the predicted groups requires more effort than manually addressing messages.

Both metrics are useful. As mentioned above, unlike the group-centric metric, the message-centric metric allows us to determine to what extent the set of future messages benefit from the predicted groups. It does not, however, inform us about how useful each group is. It is possible that a small number of the predicted groups are useful for all future messages and the other predicted groups are in fact ephemeral groups never used in the future. The group-centered metric determines the average usefulness of the groups, albeit for only one future message.

V. Results Analysis

As mentioned earlier, we chose to limit the number of constants to test when conducting our experiments. These constants are shown in Table 1.

Using these constants, we predicted groups for each of our participants and computed both the group-centered and message-centered metric for each approach, which are shown in Table 2.

These results lead us to conclude that the Google Score Graph Generation algorithm with half-life arguments of 1-week and 2-week (highlighted in green) provide the most useful groups. With these approaches, on average it took 93.3% less effort in terms of additions and deletions to convert a group to the collaborators of its closest message than it would to manually create the group of collaborators. Moreover, if all messages used the predicted groups, these best results would require 93% less effort in terms of additions and deletions than addressing those messages manually without contact groups. If our simple metrics truly capture reality, this is a very strong result, and shows the usefulness of the composition approach.

In our experiments, we also observed that the group-centered approach nearly always produces a smaller percentage than the message-centered approach. This is expected. If a group is persistent, it is likely to be close to *some* future message. However, not every message is expected be close to *some* group, as certain messages are likely to be addressed to ephemeral groups. The fact that the message-centered metric is slightly larger than the group-centered metric suggests that ephemeral groups are small in number.

Across our Google Score tests, all of our group-centered and message-centered metric results were close in value. The closeness of the results of these various methods suggests that

Table 2. Results of Group Prediction in Email

Graph Creation Approach	Time Threshold or Half-life	Group Centered Metric	Message Centered Metric
Simple Graph	N/A	7.8%	8.1%
Simple Threshold	**2 Weeks**	**18.8%**	**24.9%**
Simple Threshold	1 Month	7.8%	8.1%
Simple Threshold	2 Months	7.8%	8.1%
Google Score	**1 Week**	**6.7%**	**7.0%**
Google Score	**2 Weeks**	**6.7%**	**7.0%**
Google Score	1 Month	7.8%	8.1%
Google Score	2 Months	7.8%	8.1%

changing the half-life parameter does not have a large impact on the result of the groups produced.

One case of Simple Threshold with a threshold of 2 weeks (highlighted in red) performed much worse than any other graph generation approach we tested. This is a particularly interesting result, since its threshold is the same value as the half-life of one of best performing Google Score approaches, which is also tied for the best performing approach overall. This suggests that it is better to treat older messages as less important rather than ignoring them entirely when automatically identifying contact groups in email.

Moreover, the best Google Score approach was only slightly better than all approaches except the approach mentioned above. Among these approaches is the Simple Graph technique, which does not ignore any messages. Thus, there is limited usefulness (in terms of the metrics we used) of ignoring old past messages. One concern, of course, is efficiency. The larger the set of message we consider, the more the number of messages we must process and the more the number of edges between nodes. Our results show that after one month of data, there is a small benefit of including additional messages.

Our studies also show that the group-centric metric improves with the message-centered one – it is not the case that one approach offers better group-centered distance and worse message-centered distance than another.

Our model of user effort does not take into account the cost of naming groups. This may be an arduous task, because each group must have a unique and memorable name. The larger the number of predicted groups, the more difficult it is to choose from them. The metrics also do not penalize prediction of too many groups.

The user effort saved can be misleading in some systems as the metrics we looked at assumed both deletions and insertions. Our automatic evaluation model assumes that the users accept an initial group and never edit them afterwards. Instead, they add and remove the members generated by each group in each subsequent message that uses the group. This model has two problems. First, many email systems allow users to add to the list of recipients in a named group but not remove from them, though recent ones, in particular, Gmail, do allow deletions Second, the users will edit the initial groups, which we have not captured in our automatic model. Future work is required to either require such user intervention or assume an automatic model of user editing. For example, we can go through the series of future message, and for each message, we can find the closest committed or uncommitted group to it. If it is committed then the user effort is the number of additions to cover all recipients. If the group is uncommitted, we can perform edits to it that make it conform to the message and then commit it. In this case, the user effort is the number of additions and deletions to make it conform.

We also did not evaluate whether email messages originally contained the correct recipients. Past work has observed that at least 9.27% of users have incorrectly addressed messages [5]. This indicates that some of our participants may have incorrectly addressed messages. If our study participants had incorrectly addressed messages in the training set, we may have incorrectly predicted groups. If some messages in our test sets were incorrectly addressed, we may have incorrectly measured the effectiveness of our groups. Future work can look into techniques to remove or correct messages with incorrect recipients.

Our user study was also largely limited to university students. It is not clear that these results would apply outside this population. Social relationships may be organized differently in different settings, such as a corporate environment. If the social organization is different enough, our prediction approach may not predict useful groups. Future work may look into the application of our group prediction techniques in a wider population.

A. Performance

Finding maximal cliques is an NP-complete algorithm; thus the hybrid clique merger we used can take a substantial amount of time, depending on the size of the graph. We expect it to run in the background during lulls and only after a certain number of contacts have been added. Nonetheless, it will be useful to explore more efficient techniques to mine social graphs.

In practice, however, we found that group detection from our generated graphs did not take an inordinately long time. For example, groups were extracted from a Google Score generated graph in 3.3 seconds for a participant with over 627 email messages. Moreover, even though some participants had generated graphs with over 500 nodes, all group predictions required less than an hour.

In comparison, similarly sized Facebook graphs sometimes required over 1 week. The quick extraction of groups in the email case may be due to the sampling process we used. Since our process collected at most 2000 messages and 400 threads for each participant, these are likely not to be very dense graphs in terms of edges. With fewer messages, there are fewer chances for edges to form between nodes in the graph. Therefore, it is likely that many nodes in these graphs w not have edges between them, which then reduces the cost of finding maximal cliques. Future work may study the effect of larger email accounts on group extraction time.

VI. CONCLUSIONS AND FUTURE WORK

Our approach makes several contributions. It places group prediction algorithms in a design space with two main dimensions, domain and type of group, as shown in Table 3. We have looked at two domains – email and social networks, which use message addresses and friend lists, respectively, for prediction. We have looked at two kinds of groups – persistent groups predicted in batch without the context of an application, and ephemeral groups predicted incrementally using the context of a specific message and previously selected recipients. We have shown that it is logically possible to create a new approach for predicting persistent groups in a new quadrant of this design space by combining two existing approach in the other two quadrants.

We developed two additional novel approaches for predicting persistent email-groups: Simple Graph, Simple Threshold, and Google Score. All three approaches define

graphs that are then mined by the hybrid clique merger algorithm.

Two of the three algorithms require identification of values of parameters, time threshold and half-life & sent constant, which must be set before graphs can be created. Using actual user emails, we used cumulative distribution functions to analyze the effects of different constant values. This analysis allowed us to reduce the set of possible parameters down to a manageable, evaluable set of constants.

For our evaluation, we identified two new metrics, message-centered and group-centered, which do not require any user involvement. These metrics compute the distance between predicted groups and actual collaborators in a message, and attempt to determine both the likelihood that a predicted group will be used in a future message and that a message will use a predicted group.

Based on these two metrics, we have several interesting results. The Google Score Graph Generation algorithm with half-life arguments of 1-week and 2-week provide the most useful groups. The groups predicted by it, on average, would require about 90% less effort than no use of groups, which is a very strong result. Our results also show that simpler scoring techniques also save at least 90% user effort, and that two weeks is too small a range in the simple threshold approach.

Table 3 Design Space of Group Recommenders

	Email	Social Network
Persistent	Clique Merger + Google Score	Clique Merger
Ephemeral	Google Score	

As all of the successful approaches were based on the Clique Merger, our results imply that there is a link between group identification in email and Facebook, and possibly more generally between social networks and email.

These are only preliminary results as our work has several limitations that can be addressed by future research.

As mentioned earlier, the ability to efficiently address recipients is only one application of named groups. Our metrics do not evaluate other benefits [2] such as the ability to organize contacts. The section analyzing results discusses other limitations of the evaluation that need to be addressed in future work.

Future work is also needed to explore the link between social network and email in more depth. Do users form the same social structures and hierarchies in these two kinds of systems? Can other group-based prediction approaches, such as Gmail's recipient prediction, apply across these systems?

Finally, we have looked at relatively coarse-grained dimensions in composing research efforts. It would be useful to combine aspects of the graph mining techniques to, for instance, take the intersection or union of the predicted groups.

Despite these limitations, our work shows stark reductions in effort using the composed group prediction approach, and more importantly, provides a basis to investigate new graph mining techniques in email and the cross-pollination of group-based predictions.

ACKNOWLEDGMENT

This research was supported in part by the NSF award IIS 0810861.

REFERENCES

[1] Amershi, S., Fogarty, J. and Weld, D.S. 2012. ReGroup: Interactive Machine Learning for On-Demand Group Creation in Social Networks. *Proc. CHI* (2012).

[2] Bacon, K. and Dewan, P. 2011. Mixed-Initiative Friend-List Creation. *Proc. ECSCW* (2011).

[3] Bartel, J. and Dewan, P. 2012. Towards Hierarchical Email Recipient Prediction. *Proc. CollaborateCom* (2012).

[4] Bartel, J.W. and Dewan, P. 2013. Evolving friend lists in social networks. *Proc. of RecSys* (New York, NY, USA, 2013), 435–438.

[5] Carvalho, V.R. and Cohen, W.W. 2008. Ranking Users for Intelligent Message Addressing. *Proc. ECIR* (2008).

[6] Fisher, D. and Dourish, P. 2004. Social and Temporal Structures in Everyday Collaboration. *Proc. CHI*. (2004).

[7] Friggeri, A., G. Chelius and Fleury, E. 2011. Triangles to Capture Social Cohesion. *Proc. SocialCom* (2011).

[8] Grob, R., Kuhn, M., Wattenhofer, R. and Wirz, M. 2009. Cluestr: mobile social networking for enhanced group communication. *Proc. GROUP*. (2009).

[9] MacLean, D., Hangal, S., Teh, S.K., Lam, M.S. and Heer, J. 2011. Groups without tears: mining social topologies from email. *Proc. IUI* (2011).

[10] Mcauley, J. and Leskovec, J. 2012. Learning to Discover Social Circles in Ego Networks. *Proc. NIPS* (2012).

[11] Newman, M.W., Lauterbach, D., Munson, S.A., Resnick, P. and Morris, M.E. 2011. "It's not that I don't have problems, I'm just not putting them on Facebook": Challenges and Opportunities in Using Online Social Networks for Health. *Proc. CSCW* (2011).

[12] Palla, G., Der´enyi, I., Farkas, I. and Vicsek, T. 2005. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*. 435, (2005), 814–818.

[13] Radicati, S. and Levenstein, J. 2013. *Email Statistics Report, 2013-2017*. The Radicati Group, Inc.

[14] Roth, M., Ben-David, A., Deutscher, D., Flysher, G., Horn, I., Leichtberg, A., Leiser, N., Matias, Y. and Merom, R. 2010. Suggesting Friends Using the Implicit Social Graph. *Proc. KDD* (2010).

[15] Skeels, M. and Grudin, J. 2009. When Social Networks Cross Boundaries: A Case Study of Workplace Use of Facebook and LinkedIn. *Proc. Group* (2009).

[16] Vaidya, J., Atluri, V., Guo, Q. and Adam, N. 2008. Migrating to optimal RBAC with minimal perturbation. *Proceedings of the 13th ACM symposium on Access control models and technologies* (2008).