Network Analysis: Final Report

Inferring Social Groups from Email Archives

Sudheendra Hangal
hangal@cs.stanford.edu
(with Diana MacLean, Seng Keat Teh and Monica S. Lam)

http://suif.stanford.edu/dunbar

Abstract

We investigate algorithms to detect groups in a single users email network. Our goal is to build a practically usable tool that can mine a users email archives and based on email communication patterns, infer groups of people in the users email network. We aim to use this information for 2 purposes: 1) To seed groups for access control to data in online social networks and 2) to automatically sort and select information from different groups in social networks.

1. Introduction

While online social networks continue to enjoy runaway popularity, a big concern for many users is the relative lack of usable privacy and security features in these networks. Privacy settings on popular networks continue to be opaque; for example many people have exploited the loophole of unverified regional networks on Facebook to explore other users information (the default privacy settings allow access to anyone on the same regional network as the owner).

The relative difficulty of using privacy controls in social networking sites discourages use of these controls, and causes two problems: frequently unintentional sharing, and by extension, reluctance on the part of users to put truly sensitive information into online social networks because they do not understand exactly who has access to a piece of data. For example, sensitive information is routinely conveyed in email messages, including in email attachments; it is unlikely that such a sensitive message or document would be shared via a social network site. Email is a natural fit for such sensitive information due to the transparency of sharing: it is easy to specify exactly who the message is addressed to, and straightforward to verify.

While group-based access control is a natural and flexible technique for controlling access to resources, a particular problem in practice is the cumbersome procedure of defining groups. Group-based access control methods have largely failed to be effective for lay users for this reason. Consider the time and effort required to classify and maintain group labels for a set of 500 friends, a not uncommon friends set size on Facebook. We propose to bootstrap the process of defining groups (or friends lists) by mining email communications to automatically detect likely groups.

2. Requirements

We posit some requirements for a group detection algorithm in our context. (though we are willing to relax them in case they cannot all be satisfied at once.) Our input is a series of messages, typically on the order of several thousand, each with a recipient list. As a simplification tactic, we will ignore mailing lists (for now), though it should be possible to explode mailing lists into the constituent individuals as input to our algorithm. Our required output is a relatively small number of groups (say, 10 to 30) that can be presented to the user for inspection; the user can further tune the groups with little effort. Similar to the Dunbar number which is a theoretical cognitive limit to the number of stable relationships a person can have (the average is about 150), we speculate that there is a limit to the number of groups a person can associate with, and this number is likely to be in the few tens of groups. Some of the features our algorithm needs are as follows:

- It must support overlapping groups; i.e. a person P can be a member of multiple groups. This mirrors real life: P may simultaneously be a fellow school parent and a soccer buddy.
- The algorithm must attempt to capture the frequency with which two individuals are co-recipients, perhaps as some form of edge weight between 2 nodes.
- It must be able to create groups of people who do not all explicitly appear together in any single message. For example, if all 3-subsets of the group $\{A, B, C, D\}$ are

inferred to be valid groups, it should be possible to derive that $\{A, B, C, D\}$ is a single, cohesive group.

- The results must be tunable so that, under control of the user, a few parameters (like cohesion metrics) can be adjusted, allowing the user to play around with different settings.

We decided to implement different flavours of the algorithm and run user studies to determine which of the algorithms performed the best from the point of view of the end user.

3. Related Work

We briefly survey the most related work in this secion.

3.1 Newman's community detection algorithm

Newman's community detection algorithm [2] is widely cited as an efficient group detection algorithm; it defines a modularity metric Q based on the number of intra-group edges actually present, less the number that would be expected if edges occurred randomly. It starts with each node in its own community and then successively joins communities in pairs, choosing at each step the join that results in the greatest increase in Q. While this greedy algorithm is not necessarily optimal, it has been found to work well in practice. Newman's algorithm, as described in the paper, has a few problems in our setting: First, it is typically applied to information gathered from multiple people's data. Second, it is a form of hierarchical agglomerative clustering, which classifies people into separate groups and does not allow a member to belong to multiple groups. Finally, our experience with using Newman's algorithm found some problems with the Q-metric (see discussion later in this paper.)

3.2 Email analysis at HP Labs

Huberman et al's work [3] is representative of research in analysis of email networks. They analyzed about a million email messages sent over a 2 month period in HP Labs. They reduced this dataset to a graph with 367 nodes and 1110 edges, and extracted 66 communities from the graph. They conducted some user interviews to determine if the groups inferred were accurate. The algorithm described in their paper is based on removing edges using an approximate measure of betweenness centrality. There are at least three limitations of their algorithm. The first is that it cannot model weights between edges (edges are either present or not based on an arbitrary threshold of 30 messages exchanged, with at least 5 messages sent each way). It also models only person to person edges, thus losing information when one person sends a message simultaneously to 3 or more other individuals. Finally, it depends on a global view of the email network, which is impractical for a setting such as ours with no organizational boundaries.

4. The Algorithms

In this section we detail our algorithms to generate overlapping social groups from an e-mail corpus. In contrast to much of existing prior work, our approach focuses on extracting *overlapping* groups from *weighted*, *egocentric* social networks. Taking a weighted, egocentric approach is a good fit to the problem space of access control and data sharing, as globally oriented graphs do not necessarily capture the nuances of social ties. For example, two people may know the same group of people, but may have quite different tie strengths to the group.

4.1 Overview

Our goal is to propose a small number of groups based on the email communication pattern, each of which contains individuals who are socially similar from the user's perspective. Intuitively, we think of people as socially similar if we would share the same information with them. If we were to assign messages to groups based upon the best fit of a group to the actual recipient list, we would like to minimize the number of group members that would have access to mail on which they were not recipients.

As input, we take a single user's e-mail corpus and outputs a set of social groups whose members should be, in some sense, socially "similar" to one another from the user's perspective. Given the input corpus, e-mail headers are extracted from the user's sent mail folder (using the sent mail folder filters out a large quantity of irrelevant mail from the inbox) and used to compute the individual-group mappings.

There are 2 high level steps of our algorithm. The first generates small, highly cohesive starting groups (which could also be single individuals) in the e-mail corpus. These groups comprise of individuals with a high probability of being together in the same group; notably these groups are all sets of people who have frequently appeared together on the same message, above a threshold number of times. We explore three different algorithms for generating these starting groups. The second step takes these starting groups and merges them using Newman's algorithm. This step addresses one of the requirements we discussed earlier that the final groups may consist of individuals who have not simultaneously appeared on any message. We discuss each of these steps in turn below.

4.2 Identities and Entity Resolution

Before performing any grouping, we attempt to "resolve" all individuals in an email corpus, by identifying different email addresses or name spellings for the same individual, since both e-mail addresses and names are prone to change over time. We handle such drift by unifying and storing names and email addresses from the input email headers when either the name or email address matches a previous entry. Comparisons are case insensitive.

This simple algorithm works well in practice on our data: perhaps an egocentric network is small enough that the probability of two distinct individuals having precisely the same name is relatively low. We did occasionally observe a problem where we inferred two different identities for one person if that person different e-mail accounts with different name spelling, but it happened infrequently enough that we did not feel the need to use more sophisticated entity-resolution methods.

4.3 Step 1: Generating starting groups

Our goal in Phase 1 is to generate small sets of co-recipients that occur more than t times, where t is an occurrence threshold for the group in the e-mail corpus that renders an individual or group significant. Currently t is empirically set to 0.5% of the number of messages, capped to 5. We say that a group is frequent when its member co-occur in t or more messages.

There are three different options we consider for this step.

4.3.1 Option 1: Frequent and similar sets

We compute frequent subsets of message recipients, using a market baskets algorithm. This algorithm performs multiple passes over the input, combining ”itemsets” (in our case, message recipients) already found to be frequent.

Further to occurrence frequency (which is what the traditional market baskets algorithm deals with), we would also like these subsets to exhibit strong internal similarity in the sense that the group members appear frequently with each other, and infrequently without. For example, if A and B appear together 5 times, but A and B also appear without each other 100 times, they are substantially different from each other.

We measure similarity by extending the traditional Jaccard similarity metric to apply to more that 2 entities. We define the Generalized Jaccard Similarity (GJS) of a set of recipients as the number of messages in which all the members of the set were co-recipients, divided by the number of messages where any one member of the set was a co-recipient. If R_m is the set of recipients of a message m, then the similarity of a set S, GJS(S) is computed as:

$$\text{GJS}(S) = \frac{|\{m|S \subseteq R_m\}|}{|\{m|R_m \cap S \neq \emptyset\}|}$$

Thus, the GJS metric ranks socially cohesive sets highly. The ranking is high if set members appear frequently as co-recipients in the message corpus and if members appear infrequently without each other in the corpus. We consider a subset S similar if GJS(S) is above a similarity threshold (typically set to 0.1 in our experiments).

Note that both frequency and Generalized Jaccard Similarity are monotonic: the frequency and similarity of any superset are smaller than the frequency and similarity of a subset. Therefore, any set with insufficient frequency or similarity can be used to rule out related supersets.

We use a bottom-up algorithm to generate frequent and similar subsets. Starting with all pairs of frequent co-recipients, we compare these pairs against each message recipient list and identify the ones that cross the frequency and similarity threshold. We then create all combinations of sets identified as frequent and similar, and generate a new set of candidates, and try to find which of those are frequent and similar. This process continues iteratively till a fixed point is reached, and no more candidate subsets are left to evaluate. For fast performance, we limit the subset size to below a limit, usually about 6, because the next phase of the algorithm will combine similar groups in any case. Our algorithm is a variation on the market baskets algorithm frequently used in data mining [1].

4.3.2 Option 2: Frequent sets

Our second option is to use entire recipient lists as the starting groups. We just count the number of times the (sorted) recipient lists occur in the email corpus and use those as the starting groups.

4.3.3 Option 3: Individuals

Finally, in this option, we take the frequent individuals who occur in the email corpus and use those as a single-element groups to start the second step with. This formulation is like a traditional social network analysis that views people, not groups as nodes. However, since the second step performs hierarchical agglomerative clustering, this option rules out the possibility of the same individual belonging to different groups eventually. Note that with option 1 and 2, the same individual could indeed belong to multiple starting groups (and therefore to multiple groups reported after step 2 as well.)

4.4 Step 2: Deriving Larger Social Groups

We construct a weighted graph whose nodes are the groups $G_1, \ldots, G_n$ generated in step 1, and run Newman's algorithm to maximize modularity within this graph. Let M_i be the set of messages whose recipients include all the individuals in group G_i, and R_m be the individuals who are the recipients of message m. Then the weight of the edge between G_i and G_j is:

$$w_{i,j} = \frac{\sum\limits_{m \in M_i \cup M_j} |R_m \cap (G_i \cup G_j)|}{|M_i \cup M_j| * |G_i \cup G_j|}$$

The edge weight w_{ij}, capturing the similarity between groups G_i and G_j is always in the range [0,1]. The intuition behind this metric is as follows: if a relatively high proportion of messages including G_i also includes a significant portion (or even all) of the members from G_j, then G_i and G_j should be strongly connected. If the overlap is small, however, then the connection should be weak.

Depending on the option chosen in step 1, we will refer to the three variations of the algorithm as *A1:* (Newman-

	Avg.	Max	Min	Std. Dev.
Messages analyzed	5147	14988	699	4229
Timespan (months)	49	65	15	18
Unique Individuals	570	970	52	351

Table 1. Email statistics in user study

Frequent Subsets), *A2:* (Newman-Frequent Sets) and *A3:* (Newman-Frequent Individuals).

4.5 Implementation

The algorithms we discuss are implemented as part of Project Dunbar, an email mining project developed primary by the author. The CS322 project implemented automatic group detection from email messages and is integrated into Dunbar. The author was the primary developer for the algorithms discussed above; Diana MacLean provided help in refining the algorithms as well as for implementing the parts in R. We used the igraph package available in R to run Newman's algorithm.

5. User Study

We ran a user study of the effectiveness and promise of our approach with 14 users who were above average users of email and Facebook for social networking activities and the sharing of personal data. The setup of the user study required users to run our analysis on folders they selected from an IMAP/POP email account (or local files in mbox format). We then asked them to evaluate the groups generated by our three algorithms and how accurately they reflected their natural social groups.

5.1 Results

Of the 3 algorithms that we have provided to infer a user's social groups from email, 9 users found *A1* (Newman-Frequent Subsets) to be the most accurate, while 1 user chose *A2* (Newman-Frequent Sets) and 4 users chose *A3* (Newman-Frequent Individuals). All participants thought the groups identified were very cohesive, especially for *A1* and *A3*. Most people were positively surprised by the accuracy of the identified groups. We heard comments like *"you got my core group of friends there"*, or *"you identified my family"* or *"there's the board of my company!"* For most users, between 20 and 40 groups were reported.

The most common complaint we heard was that occasionally a set of people who were strongly connected among themselves, were grouped with another person who was peripherally connected to the group.

We also surveyed users on their use of Facebook's *Friends List* feature with which they can filter their Facebook social feeds or limit access to their data. The drawback to this feature is that it requires users to manually create and maintain these friends lists as their social relationships evolve. Only 3 users were aware of this feature, and virtually all our users agreed that the manual work involved in managing these lists would be a significant deterrent to their using the lists.

6. Learnings

A frequently used metric for detecting communities in networks is the modularity metric, as defined by Newman [2]. The modularity metric Q for a grouping is:

$$Q = \sum_i (e_{ii} - a_i^2)$$

e_{ij} is defined as one-half of the fraction of edges that connect vertices in groups i and j, when i and j are different, and as the fraction of edges that fall within group i when $i = j$ (i.e. no one-half factor for intra-group edges). a_i is defined as $\sum_j e_{ij}$. In other words, a_i is the fraction of edge-ends that lie within group i. Therefore, if the network were rewired randomly, while maintaining each node's degree, the expected number of edges within group i would be a_i^2. The greedy Newman algorithm successively combines the 2 existing groups that give the best increase in Q, continuing till the Q-value stops increasing. The same formula applies in case of weighted edges, except that the e_{ij} matrix represents the weighted fraction of edges.

The formula above has a problem in that it will always draw in "whiskers" into otherwise strongly connected groups. This is because it does not correctly assign probabilities for groups containing only one node. Consider 2 nodes A and B with a large edge weight x and a node C connected to B with a weight of 1. The Newman algorithm in this case will first merge A and B. After this step, the Q-metric for the grouping where group 1 is $\{A, B\}$ and group 2 is $\{C\}$ is:

$$Q = e_{11} - a_1^2 + e_{22} - a_2^2$$

Given that

$$a_1 = e_{11} + e_{12} = \frac{x}{x+1} + \frac{1}{2(x+1)} = 1 - \frac{1}{2(x+1)}$$

and

$$a_2 = e_{22} + e_{12} = 0 + \frac{1}{x+1} = \frac{1}{x+1}$$

$$Q = \frac{x}{x+1} - (1 - \frac{1}{2(x+1)})^2 + 0 - (\frac{1}{x+1})^2$$

leading to a Q value that is always slightly less than 0. Therefore, the Newman algorithm will merge C into the group $\{A, B\}$, because the grouping $\{A, B, C\}$ has a Q value of 0.

In reality, a_1, the expected fraction of intra-group edges in group 1 should be $\frac{x}{x+1}$ and a_2 should be 0, assuming self-edges are disallowed. Therefore the modularity of the original grouping before C is added to $\{A, B\}$ should also be 0. The problem with the Q definition is that it does not account for groups with only one node. Indeed, if we add shadow nodes for each original node in the graph and connect the shadow node only to its corresponding original node, we find that the groups $\{A, B\}$ and $\{C\}$ remain disjoint.

7. Software Release

The software for this project is available for testing by the interested reader at http://suif.stanford.edu/dunbar. Users can login to an email server, select some folders and then click on the groups button to see the results of the 3 algorithms compared in this paper. To alleviate privacy concerns, users can can download the software and run Dunbar in the privacy of their own machine. (R must be installed on the local machine for the Newman algorithm to work).

8. Conclusions and future work

We implemented three algorithms for group detection in ego-centric networks and demonstrated with user studies that a fairly simple algorithm was able to derive groups from users' email corpora. The variation that works best is one that derives small frequent and similar subsets and then applies Newman's algorithm to form larger groups. We are continuing to refine our algorithm to capture not only the accuracy of the formed groups, but also ensure that the derived groups are representative, i.e. cover as large a fraction of messages and recipients as possible.

References

[1] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 1993.

[2] M. E. J. Newman. Fast algorithm for detecting community structure in networks, September 2003.

[3] J. R. Tyler, D. M. Wilkinson, and B. A. Huberman. *Email as Spectroscopy: Automated Discovery of Community Structure within Organizations*, pages 81–96. Kluwer, B.V., Deventer, The Netherlands, The Netherlands, 2003.