Towards Evolutionary Named Group Recommendations

Jacob W. Bartel[1]; Prasun Dewan
University of North Carolina at Chapel Hill
bartel@google.com; dewan@cs.unc.edu

This is a post-peer-review, pre-copyedit version of an article published in Computer Supported Cooperative Work (CSCW). The final authenticated version is available online at: https://doi.org/10.1007/s10606-018-9321-5

Abstract. When sharing information, a common tactic for reducing the cost of choosing recipients is to form *named groups* of users. These groups are then selected as recipients in lieu of or in addition to users. However, keeping named groups up to date is a difficult and error-prone task when conducted manually. Past schemes automating this task make different tradeoffs and can be distinguished based on several factors including the types of named groups they consider, whether they evolve a specific group or a set of multiple groups, and how integrated they are with techniques for predicting initial groups. We analyze these approaches and identify a design space of potential evolutionary approaches. Using this analysis, we introduce a novel approach for automatically suggesting a sub-type of evolution, evolutionary growth. This approach (a) requires no prior knowledge of which groups change, (b) composes, and therefore interoperates, with an existing engine for recommending named groups, and (c) extracts groups from the social graph of multiple types of applications regardless of whether the graph are explicit or derived implicitly from message communication. Our evaluation considers social graphs created using explicit and implicit connections, and identifies the conditions under which the approach outperforms baseline techniques.

[1] Now at Google LLC

1 Introduction

Many types of collaborative systems require that users choose other users with whom they should share some piece of information. The form of the shared information varies; moreover, in some systems information is pushed to the sharers while in others while in other it is pulled by them. Distributed repositories require that users choose who has access to repositories or files; instant messaging systems require users to direct messages to particular users; email messaging systems require users choose email addresses as recipients; and online communities often require users to choose in which forums to post. In all these systems, possessors of some information face the *user-selection problem* – the problem of selecting the users who receive the information or can retrieve it. This is a problem because it is possible for users to make mistakes when addressing selecting users, which in turn, indicates there is a sense of correctness.

Users have many goals when sharing information such as building social capital (Burke et al. 2011), finding recommendations for services (Daly et al. 2014), or solving problems (Burke et al. 2011; Pal et al. 2012). Users incorrectly address the user-selection problem, either inadvertently or purposefully, when their selection or omission of a user is not consistent with their sharing goal. Incorrect user-selection can lead to significant, adverse effects. For example, Congresswoman Michelle Bachman was accidently addressed in a private email critical of her (Molloy 2011), many Facebook users inadvertently granted access to more advertisers than that should have been allowed (Soghoian 2008), and participants in study did not share possibly useful health information on Facebook because it was too cumbersome to correctly choose privacy settings (M. W. Newman et al. 2011).

A reason user-selection is a problem is that the selector must remember all users who can receive information and determine which of these potential users should be selected. It is a subclass of a more general memory problem identified by (Miller 1956). He found that that subjects in a study could only remember 5-7 elements from lists of digits, letters, or words and could at most correctly specify a portion of binary features of phonemes. He also identified an approach to reduce the difficulty of this problem. Subjects could remember more of each type of item after they were grouped. For example, users could only remember a limited number of characters individually, but they could remember more characters that were grouped into words. Furthermore, by deciding based on groups of options rather than individual options, the user may need to make fewer binary decisions about whether to select an option.

The idea to group entities to reduce cognitive load has been used in computer-science to reduce the user-selection problem by allowing users to sort contacts into *named groups* (e.g. “Coworkers”, “Family Members”, “Friends”), so selections can come from the set of named groups which is likely to be smaller than the set of all

possible users (Bacon and Dewan 2011; J. W. Bartel and Dewan 2013; Lampson 1974; Reeder et al. 2008; Shen and Dewan 1992). Such capabilities have been implemented in many currently used systems, such as Facebook, Google+, Twitter, and email clients

To leverage these named groups, users must first incur the cost of identifying and creating them. One way to reduce this overhead is to automatically predict or recommend the creation of these named groups, which we refer to as *foundational named group predictions*. Recent research has investigated foundational named group predictions in a variety of contexts, such as role mining (Vaidya et al. 2008), community detection (Palla et al. 2005), ego-network identification (Backstrom et al. 2006; Bacon and Dewan 2011; McAuley and Leskovec 2012), and email communication (Fisher and Dourish 2004; MacLean et al. 2011).

It is not sufficient create groups with all the correct members at a specific time, either through recommendations or manual grouping. Past work has observed that the state of a social networks is susceptible to temporal effects for several reasons. Social graphs can change when individuals move into different neighborhoods and interact on associated forums (Daly et al. 2014). Targeted communication over time can lead to the creation or improvement of relationships, which can lead to new edges in the social graph (Burke et al. 2011). Similarly, both social capital and relationships have been observed to change over time in a study of Facebook activity (Burke et al. 2011), and, in fact, trust has been observed to change overtime among individuals in a single IM thread (Kalman et al. 2013),

Because groups have been successfully inferred based on properties of the social graph and relationships, changes to these states imply change to the groups as well. This implication is supported by previous work. It has been observed that groups tend to be dynamic with new members being added and old members being removed over time (Roth et al. 2010). It is therefore important for named groups to evolve over time to prevent them from becoming stale. Keeping these groups up to date with the social graph and efficiently organized requires additional costs. These include the costs of identifying and creating new groups as well as modifying existing groups. Again, it is possible to reduce these costs with automatic evolution predictions or recommendations, which we refer to as *evolutionary named group predictions*. Past work has developed initial approaches for making predictions in this category (Amershi et al. 2012; Backstrom et al. 2006; Vaidya et al. 2008), but has also stated that this is an open problem for groups extracted from social graphs (Amershi et al. 2012; MacLean et al. 2011). Namely past work has required either (a) that recommendations be made for a single group or individual, despite the fact that users often have multiple groups or graphs changed by more than one users, or (b) that predictive models take, as input, a collection of multiple features about both individuals and groups, which limits approaches to applications that have access to features compatible with past techniques.

This paper presents a general framework for understanding named group evolution and classifying existing work in this area. The framework is described by first presenting a design space of techniques for automatically recommending named groups, and then using this space derive a design space of possible approaches to evolve named groups. This framework is then used to motivate, explain, and evaluate an application-independent approach that relies only on growth rate of the social network and an unweighted social graph to adding new members to evolving named groups. Our evaluation considers named groups in both email and social networks, which are extracted from social graphs created from message communication and explicitly specified connections, respectively. The paper ends with conclusions and directions for future work.

2 Design Space of Named Group Recommendations

As mentioned previously, the creation of named groups come with a cost. Past works has sought to address this problem by automatically identifying groups in a variety of contexts.

2.1 Mined Data: Assigned Permissions vs. Social Graphs

Named group recommendation techniques must mine existing data about users to extract the groups. These techniques can be classified about the nature of the data mined.

Role-based access control extracts these groups, called roles, from user-permission assignments. Its goal is to group users, or subjects, who have been given similar permissions so that the cost of specifying future permissions is reduced. It has employed both a top-down approach, where existing groups are divided into functional units that share the same access rights, and a bottom-up approach, where existing subject-permission assignments are aggregated into roles (Vaidya et al. 2008).

An alternative approach, which does not need to be seeded with user-permission assignments, detects groups, called communities, based on high connectivity or similarly profiled in a social network. Community detection has spanned a broad range network types, which include two general categories *co-occurrence graphs* and *social graphs.* Co-occurrence graphs are constructed based on nodes that co-occur together, and community detection in this area includes work in, such as citation networks (Palla et al. 2005), word free associations networks (Palla et al. 2005), and biological networks (Palla et al. 2005) Social graphs focus on the indicated social relationship between users, and there is a rich amount of community detection work in this area as well (Backstrom et al. 2006; Bacon and Dewan 2011; McAuley and Leskovec 2012; Palla et al. 2005).

2.2 Overlapping vs. Non-Overlapping Groups

Most of the previous community-detection work has detected non-overlapping groups (Palla et al. 2005). However, groups for controlling who to share information with tend to overlap. For example, a computer science professor may be a member of multiple named groups such as a "Research Group" and "Comp 101 Lecturers". To address such cases, other work has identified alternative community-detection approaches for overlapping groups in general networks (Palla et al. 2005).

2.3 Global vs Ego Groups

Community detection approaches in social graphs categorized based on whether they identify global groups for all users or whether they identify groups for each individual user. The former finds user-independent groups in the entire graph of a social network. The latter identifies groups for an individual user based on the user's ego social networks (Bacon and Dewan 2011; Friggeri et al. 2011; McAuley and Leskovec 2012). An ego network or graph is a subgraph of the global social graph only contains vertices one edge from a chosen individual. In other words, it contains only a user's friends or connections as nodes. Ego groups are related to roles in that they care both used to assign the same permission to a set of users (Bacon & Dewan, 2011). As mentioned above, mining of these groups relies on social graph connections and existing permissions, respectively.

2.4 Implicit vs Explicit Connections

The initial work on ego network mining techniques (Bacon and Dewan 2011; Friggeri et al. 2011; McAuley and Leskovec 2012) relied on social graphs in which the user connections were explicitly indicated by the owner (user for whom they were created) through, for instance, friend relations in Facebook. This work has also been applied to email by creating social networks implicitly from email communication. In these networks, nodes represent email recipients and edges are formed between two nodes A and B based on the messages that recipients A and B co-occur in. Such mining has been used both for a pure analysis of social graphs (Fisher and Dourish 2004), predicting groups that can be used in future messages sent by the owner (MacLean et al. 2011), and predicting ephemeral, ad-hoc groups that consisting of the recipients of a single future message (J. Bartel and Dewan 2012; MacLean et al. 2011; Roth et al. 2010).

2.5 Summary of Design Space

In summary, named groups can be mined from existing permissions or social-graph connections, leading to role and social-group recommendation, respectively.

Social-group recommendation can consider all connections or only those in which a particular user is involved, leading to community and ego-group recommendation respectively. Ego-groups are like roles in that can be used to restrict sharing to a selected set of users. The connections from which they are derived can be explicitly specified by their owner or implicitly derived from email involving the owner.

3 Framework for Understanding Named Group Evolution

The general problem of evolving or maintaining system data structures has been addressed in several domains including text-editors (Elliot et al., 1971), program repositories (Tichy, 1985), and persistent objects (Banerjee et al., 1987). This work has identified how to show differences between the original and changed data structure (Elliot et al., 1971), reconcile concurrent evolutions of the data structure (Tichy, 1985), and evolve the data structure in response to changes to some other data structure on which it depends, such as an object schema (Banerjee et al., 1987). Here we consider evolving named groups in response to changes on the underlying factors on which they depend.

If manually-creating and recommending named groups are rich problems, with many possible, alternative approaches, then the evolution or maintenance of these groups is even more complex problem. For each existing technique used to create named groups, a variety of approaches for evolving them can be developed. However, because of the infancy of this area, very few of these approaches have been developed or identified. Previous work on named groups have pointed out the need for research on named group evolution (Amershi et al. 2012; Bacon and Dewan 2011; MacLean et al. 2011).

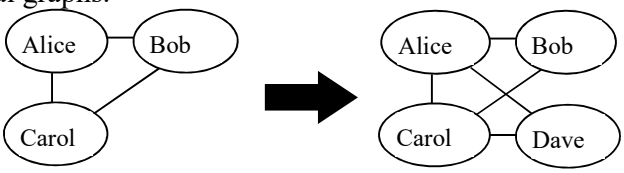
To better identify the problem of group evolution and as a step towards making such work a first-class research area, this section focused on defining the problem and an associated design space of existing and possible approaches to address the problem.

3.1 Why Evolution Occurs

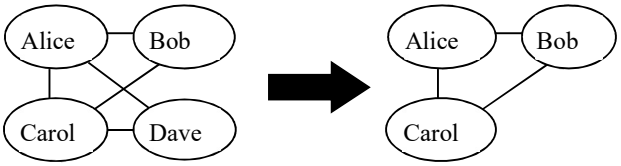
Named groups, whether created manually or through recommendations, are linked to or influenced by structures in a user-graph. In role-based systems, the graph indicates permissions a user in relation to protected objects, while in other systems, it indicated relationships between users, specified explicitly or extracted implicitly. Therefore, it is first helpful to identify how user graphs can evolve.

There are three ways this can happen: members can be added (Figure 1(a)), members can be removed (Figure 1(b)), and connections can be changed (Figure

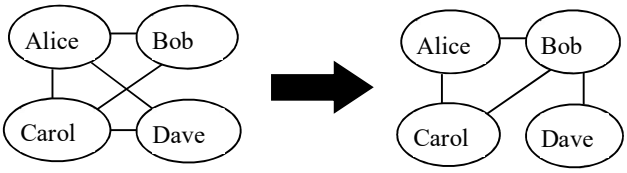
1(c)), any of which can occur concurrently. Each of these patterns of user graph evolution can occur for a variety of reasons. Moreover, the driving forces behind these types of evolution in graphs can change based on whether they are social or permission graphs, global or ego-centric graphs, and connections are explicit or implicit in social graphs.



(a) A social graph adding members



(b) A social graph losing members



(c) A social graph with changing connections

Figure 1. How a social graph can evolve

If an ego-centric graph has explicit connections, owners of graphs specify that they have a relationship with individuals, such as marking other users as friends in Facebook, professional connections in LinkedIn, or co-authors in ResearchGate. In these systems, social graphs typically add members because owners have newly specified that they have a relationship with some other users. Users may specify these relationships for a variety of reason. For example, they may wish to communicate with someone they have not beforehand, they may be specifying a relationship that started elsewhere (e.g. at an in-person meeting), or they may want to increase the size of their social graph to have higher social status. Similarly, social graphs in these systems often lose individuals, because owners specify that they no longer have a relationship with some user. Again, this can occur for a variety of reasons. For example, a user may not be interested in further communication with these other users (e.g. removing high school friends after graduating), or users may no longer wish others to associate them with this other

individual. Finally, changing connections occur when users other than the owner add or remove individuals from their ego social graphs.

In an ego-centric graph with implicit connections between users, connections are based on relationships that are inferred from shared messages (Fisher and Dourish 2004; MacLean et al. 2011; Roth et al. 2010). In these systems, individuals are added to the social graph only after they are seen together in a message for the first time. In some implicit graphs, edges are only added and never removed, while, in others, an individual or edge between two users is only included if there is a sufficiently recent message and/or large number of messages representing the user or connection.

A role-based system can evolve for similar reasons: new users may be given permissions, and permissions of existing users can be revoked or amplified in response to changing roles. In a community-based system, users may wish to enter or leave communities based on evolving interests.

Therefore, in all systems, user-graphs can evolve by adding new individuals and connections, but only a portion of such graphs allow the removal of individuals and connections. One way to grow this area is to focus on a subset of these kinds of changes and to increase or change this subset in future work. As we see below, this is the research model adopted in this paper.

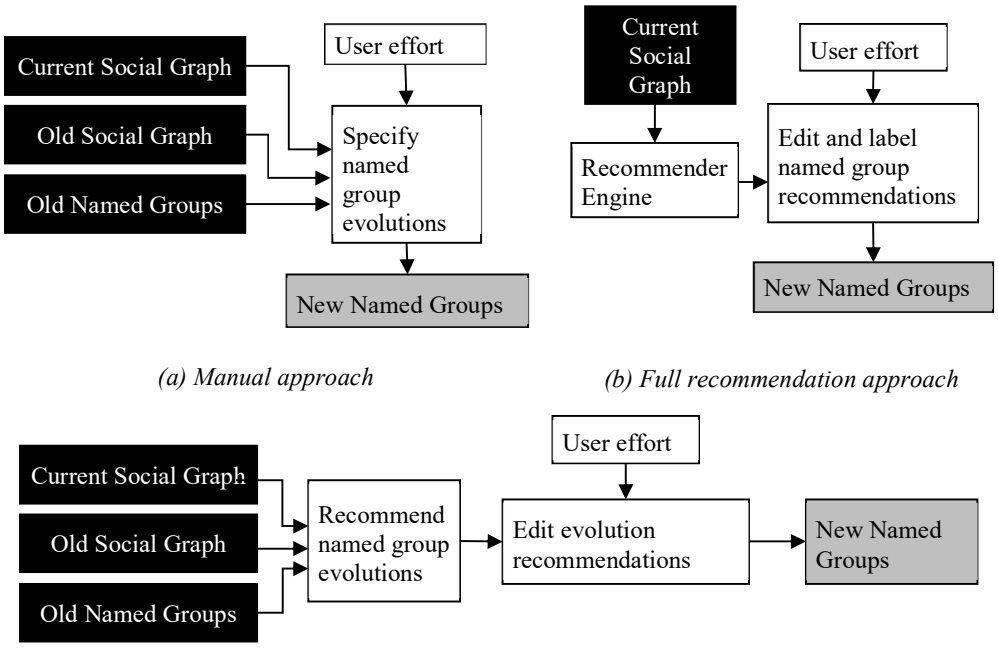
3.2 How and If Evolution is Automated

When initially composing named groups, there are only two practical approaches to consider: fully-manual and semi-automatic. Both approaches have been applied in practice, where a fully-manual approach requires no automation or predictive system, and a semi-automatic approach uses a foundational recommendation tool to propose a set of groups, which are then corrected by the user to create final lists. We do not consider an approach that assumes that an automated system can perfectly predict a useable set of named groups, which is not possible with the state-of-the-art. A machine-learning/data mining approach cannot be guaranteed to have no false positives or negatives, and named group prediction is no exception.

In the case of evolution of these groups, there is a broader set of categories based on whether the initial groups were created automatically or not and if and how tools are used for group evolution. We have identified four of these approaches, three of which are illustrated in Figure 2:

1. *Manual evolution* - As other users are added to or removed from the user graph, the user evolves manually or automatically generated named groups using no tools. This is shown in Figure 2(a).
2. *Full Recommendation* – A named group foundational recommendation tool is used to recommend a whole new set of named groups, which are then manually corrected. This is shown in Figure 2(b).

3. *Change Recommendation* – A tool recommends which named groups should evolve and how, and the user edits these recommendations. This is shown in Figure 2(c).
4. *Fully Automated Evolution* – A tool recommends the exact which named groups should evolve and how with no input or edits from the user. This approach is not yet possible with the state of the art.



(a) Manual approach

(b) Full recommendation approach

(c) Change recommendation approach

Figure 2. Approaches to named group evolution

Of these approaches, the manual approach has the possibility of being the most precise because it takes all a user's intentions into account. However, it requires significant effort if the graph has undergone significant amount of change in the form of changes to its nodes and connections.

Two types of foundational named group recommendation can be used in the full-recommendation approach, member suggestion and group creation (J. W. Bartel and Dewan 2013). Member suggestion proposes new members for a specific group that is input to the tool by the user, which may be empty in case of creating a new group. On the other hand, group creation suggests a full set of named groups. Each of these approaches has its pros and cons in the context of named group evolution.

Member suggestion does not identify specific new groups, and requires the users identify when a new group should be created. Moreover, it requires the manual overhead of specifying each user group to the tool, which is wasted if that group requires no change. Group creation does not have this disadvantage as it only present users with named groups that are recommended to change.

To illustrate, consider a scenario with two groups A and B, where A changes only with the adding of two members and B remains unchanged. In this scenario with member suggestion, the user may first unnecessarily ask for recommendations for B. Group creation can recommend new groups and present the user only those groups that are recommended to change. Thus, when there are no or few unchanged groups, group creation can require fewer steps.

However, because of group creation's batch nature, it cannot use feedback to adjust recommendations. In comparison, member suggestion's iterative nature allows feedback throughout the recommendation process, which can be used to improve future recommendations. If a user overrides an initial wrong suggestion, member suggestion can correct its later predictions. Group prediction would require the user to correct both predictions.

Even though full recommendation cannot use feedback in evolution, assuming such an engine is effective for recommending initial groups, which has been shown to be true in past work, it could work better than the manual approach. Full recommendation could introduce new groups or changes that the users failed to identify and reduce the cost of evolutions the users did identify. Moreover, such recommendations could be easily manageable if the number of groups is small and may require significantly less effort if the makeup of the groups has changed by a relatively large amount. However, full recommendation may reintroduce previously rejected recommendations – a cost not incurred in the manual approach. Thus, it is difficult to predict which of these approaches will require less effort. If the social graph has changed by a small amount, the effort required to manually add the members might be less than that required to correct and label the predictions of a named group recommender.

The problem with full recommendation is that its predictions do not consider the ways in which the user has previously specified named groups either manually or by manipulating recommendations. This problem, of course, motivates the third approach of recommending changes to existing named groups based on the previous and new social graphs (Figure 2(c)). We refer to this approach as change recommendation. It attempts to combine the positives of the manual and full recommendation approaches. Just as with full recommendation, the user must accept, edit, or reject the recommendations, but change recommendation may reduce unnecessary user effort by basing its recommendations on previous user actions.

3.3 Composed vs. Integrated Change Recommendation

There are at least two approaches to creating a change recommendation engine, the composed and integrated approaches. The former uses the results of a full (group) recommendation engine, which it treats as a black box, feeding the current social graph to the engine. It then matches and merges the results of the recommender engine with existing named groups to generate recommended evolutions of the

named groups. As a result, it can be interoperated with any such engine – hence the name composed. The integrated approach, on the other hand, makes its recommendations from scratch, building new evolutionary recommendations based on the existing groups and the social graph in its current and previous states. The composed approach leverages the previous research in recommender engines. However, an integrated approach may generate more precise evolutionary recommendations by basing recommendations on the changes rather than matching groups.

3.4 Characterizing Existing work

Approaches have been used to predict how groups should update in a variety of systems. such as role-based access control (Vaidya et al. 2008), LiveJournal (Backstrom et al. 2006), and DBLP (Backstrom et al. 2006).

In one approach targeted at Twitter, evolutions were limited to two named groups (Hannon et al. 2010), but observations in other work indicate users often will create more than two named groups (Amershi et al. 2012; Bacon and Dewan 2011).

A member-suggestion approach targeted at Facebook supported more than two named groups (Amershi et al., 2012). It displayed a ranked list of all new users to be added to a group for which new suggestions were requested by a user. This list was ranked, using Bayesian classifier, according the probability that each non-member of a group should be added to the target group based on the shared features of the non-member and the existing members of the group. This approach integrates foundational group recommendation and evolution in that it can be used to both create new groups and evolve them. As mentioned above, member suggestion is limited in that users must know which groups to add members to, whether that be the new group they are creating or the existing group they are evolving, and users will have to scan many potential users from which only a few may be selected. Both limitations require significant cost to the user when evolving groups and may frustrate users.

Other works have identified similarity between an individual and an existing individual in a social graph based on edges between them. Such approaches often use geodesic distance, or the length of the shortest path between two vertices in a graph (M. E. Newman 2008). However, this distance is between two individual vertices, and it is not a distance between a single vertex and groups of vertices.

Some role-based systems have focused on predicting evolutions of groups of users with the same or similar permissions while minimally changing the membership of any existing group using the Jaccard Coefficient as a measure (Vaidya et al. 2008). However, group creation and evolution are often independent of rights associated with their members, and may instead be focused on other properties, such as observed social groups observed outside the system (e.g. coworkers). Furthermore, optimal evolved groups may differ significantly from

their initial state such that minimal changes to membership may lead to incorrect and unhelpful groups. Finally, this approach relies on both existing and new users being assigned permissions.

Other work has predicted the likelihood that an individual joins a community (Backstrom et al. 2006). However, this implies that an individual has control over whether they may join a group or not. In ego-centric systems, the individuals have no control and often no knowledge of the groups of an owner, and thus such likelihood predictions are not applicable.

As a general problem, predicting evolutionary growth of existing groups can be classified as a sub-problem of clustering. Each named group is equivalent to a cluster and new members of the social graph can be considered as a set of new clusters that are merged with the most similar existing clusters. However, the most common and successful general approaches in past work, e.g. k-means, determine similarity between a new node and existing clusters based on an aggregated multi-dimensional feature vector from the current members of the existing cluster (Han & Kamber, 2011). These features include information beyond unlabeled edges between nodes, such as shared interests or geolocations. Such features come with additional costs in algorithmic development. Specifically, for each new application of an evolutionary approach, features must be identified and tested. Moreover, helpful features must be accessible to the algorithm implementation, which requires that (a) the implementation is included as a part of a system that has access to such data or (b) users explicitly grant access to the features. Neither is true in many cases for a variety of reasons such as organizational complications or privacy concerns.

Such features have been used to make predictions about named group evolutions in two approaches surveyed above - Facebook (Amershi et al. 2012) and Twitter (Hannon et al. 2010). Given that past work in collaborative systems has successfully made foundational named groups recommendations without any additional information beyond an unweighted social graph (Bacon and Dewan 2011; Friggeri et al. 2011), our work explores if such an approach is possible in evolutionary information using similarly limited information. It builds on past work by (a) evolving groups that are not restricted in number, (b) not requiring users to ask for recommendations for specified group, (c) not depending on permission assignment or user properties, and (d) not requiring any additional information other than an unlabeled social graph and the edges and nodes that have been added to the social graph.

4 A Novel Change Recommendation Approach

4.1 Scope

Consistent with the model of gradually the broadening the scope of evolutionary recommendations, our approach overcomes some of the limitations of previous work. We describe here the scope of our work.

We focus on change recommendation as it can overcome problems of both manual evolution and full recommendation. Our goal was to develop a method for making change recommendations without requiring the user to select the named groups to evolve. Therefore, we focus on a change recommendation engine that works in batch by suggesting changes to a set of named groups input to the engine. These changes are based on a set of heuristics, discussed below.

Our heuristics are not designed to support arbitrary changes to groups. A mentioned above, in all systems, whether they support explicit or implicit connections, social graphs can evolve by adding new individuals and connections, but only a portion of such graphs allow the removal of individuals and connections. Social graphs formed from implicit connection typically add nodes and edges based on messages between entities or co-occurring entities (Backstrom et al. 2006; Fisher and Dourish 2004; Roth et al. 2010). In many of these implicit graphs, edges are only added and never removed, because new messages or co-occurrences do not cause the removal of nodes or edges from the graph. Furthermore, in the area of explicit social graphs, the removal of nodes or edges implies that users manually remove other users from their ego networks (e.g. unfriending in Facebook). Past work has observed these removals are rare, because it is viewed as an extreme relationship management strategy due to the social costs. Users may appear more stuck-up or lose other ties because of the terminated connections, and users instead tend to block or stop following updates, which does not affect nodes or edges in the social graph (Peña and Brody 2014).

To match these common conditions, our heuristics and evaluation only target social graphs that evolve by adding new individuals or connections. The goal of our work is to recommend how groups should grow based on growth of the social graph. However, if there are conditions in which users are removed from the social graph, our composed approach will respect these changes in the social graph. Removed nodes, by definition, are not included in the unevolved groups. Furthermore, our approach will only recommend the addition of users that are currently in the social graph, meaning that users removed from the graph will not be included in recommended evolutions. Thus, our approach is consistent with the design principle of making the common case efficient while making the uncommon case possible.

This focus on growth of groups fits with the types of recommendation that would be made by the member suggestion approach of past work but works in batch.

Recall that in these approaches, evolution of a previously specified group can be recommended by suggesting which other individuals to add to the chosen group. This restriction allows the approach to focus on specific types of evolutions, but still apply to a wide variety of conditions. We leave it to future work to go beyond this initial work to recommend the removal of users from groups.

As mentioned above, a change recommendation engine can be integrated or composed with a foundational-recommendation engine, which in turn, can be integrated or independent of the collaborative application such as email or Facebook. To make our design independent of a particular application and foundational-recommendation engine, we chose the interoperating composed approach.

The key issue in a composed approach is how it maps existing named groups to the recommended one(s). Some recommendations may not be matched to any original named group. To be effective, a composed approach must correctly identify whether a recommended group should be matched with any previously existing groups. Furthermore, a composed approach may determine whether unmatched recommendation should be suggested as newly created groups or ignored as poor groupings. As an initial attempt at such an approach, we identify only which recommendations should match with previously existing groups and to ignore any groups that were not matched. Future work may extend our work by determining which unmatched groups should be used to recommend the creation of new groups.

In general, there are four kinds of such matches: one-to-one, one-to-many, many-to-one, and many-to-many. In one-to-one mapping, each original named group matches with a single valid recommendation, and vice versa. In one-to-many, a single named group may be mapped to multiple valid recommendations, and in many-to-one, multiple original named groups map to one valid recommendation. Many-to-many is the most general case, combining many-to-one and one-to-many. Again, one way to incrementally grow the research in this area is to focus on a subset of these mappings, and gradually increase/change this subset. This is the model we adopted. Our work, as well as the example we have presented so far, focuses on one-to-one mappings.

4.2 Diffing vs Matching

The challenge of a one-to-one approach is to separate valid recommendations from spurious recommendations and to map each valid recommendation to an original named group. To do so, our scheme must determine the closeness between the elements in the two sets of groups. This matching can be tied to action of performing diffs on files or folders. In diffs, files or folders are compared by determining what needs to be added or removed from one file to make it equal to another. Similarly, in the named group evolution case, the recommender is trying to find what needs to be added or removed from a result of the recommender engine

to make it equal to an old named group. However, a diffing algorithm is given the previous and new versions of a file or folder, while in our case, we are given sets of old and new versions and must filter out spurious new versions and for each old version determine the new version.

Therefore, we compare each of the old named groups with each of the recommendations to determine the recommendation that is closest to it, which we refer to as the valid recommendation of the named group.

4.3 Matching Heuristics and Group Closeness

We use the following heuristics to determine if a recommendation, R, is valid for an original named group, G:

1. R has the vast majority of the elements of G.
2. R has few members of the original social graph that were not in G.
3. The number of new individuals (individuals not in the old social graph) in R matches the growth of the social graph.

Based on these heuristics, we have defined a closeness metric, which is used in our algorithm. If closeness between a named group and a recommendation is not within a certain threshold, the recommendation is not an evolution of the named group.

To measure this closeness of a recommendation at time t_k to an old named group from time t_{k-1}, the composed change recommendation engine separates the recommendation into an old membership and a new membership, where old membership denotes members that were a part of the social network before time t_{k-1} and new members where those that joined the social network after time t_{k-1}. Based on the heuristics discussed above, we do not expect the old membership to change, and therefore expect the optimal case to have 0 additions and 0 deletions when transforming the old membership of the recommendation to the old named group.

In the case of new membership, the composed change recommendation engine expects the named group to grow similarly to the social graph. Therefore, if p is the ratio of new to old members in the social graph, we expect the number of new members in a valid recommendation to be p*|old named group|.

Given these expected numerical values for each possible pair of old named groups and recommendations we have two vectors: $v_{expected}$ = <0, 0, expected new members> and v_{actual} = <adds, deletes, new members>. It is then possible to assign a numerical value for closeness as the Euclidean distance between the endpoints of the two vectors. The smaller the closeness measure, the more likely it is that the groups match.

To illustrate the computing of closeness, consider the following example. At some point between when the named groups were created (t_{k-1}) and the current time (t_k), Joe's company hired two new people, Greg and Hal, and Joe's social graph grew by 1/3. Because of this overall growth, the composed change recommendation

engine would expect Joe's named groups to grow at the same rate, adding one new member – either Greg or Hal – each. Also in the case of our example, the recommender engine found the recommended named groups A={Alice, Eva, Frank, Greg}, B={Alice, Bob, Frank, Carol, Greg}, and C={Alice, Carol, Frank, Greg, and Hal}. Because the old membership and new membership of A exactly matches the membership and expected growth of Research Group, respectively, the value of closeness(Research Group, A) is 0. On the other hand, because the old membership of B includes one more member than Social Group, closeness(Social Group, B) is 1. Moreover, because the old membership of C requires 1 addition and 1 deletion to reach Social Group and the new membership of C is 1 greater than the expected growth of Social Group, closeness(Social Group, C) is $\sqrt{3}$.

4.4 Multi-Round Matching with Adapting Closeness Thresholds

The next step was to appropriately define a threshold of closeness for matching old named groups to recommendations. With low thresholds we found few, but precise, matches. In our example above, with a low threshold limit, the matcher is only able to match recommendation A to the Research Group, which means no evolution is recommended for the Social Group. In the case of high thresholds, we found a high number of matches, but many of these were incorrect or were not matched because they were not one-to-one mappings. For example, if we used high thresholds the above example, we may match A to Research Group and both B and C to Social Group. This means that not only is C matched even though it contains both Greg and Hal rather than just one of them, but the Social Group now has a one-to-many mapping, because it is matched to both B and C. This means again the composed change recommendation engine cannot recommend an evolution for Social Group.

Therefore, low thresholds tend to yield very few matches, but those matches tend to be correct. On the other hand, high thresholds yield many matches, but poor matches obscure good, one-to-one mappings provided by the low thresholds. Intuitively, a good approach would select matches at different threshold levels. In this way, the low threshold matches would be selected first, before they are obscured by high-threshold matches. Then matches with a higher threshold would be allowed to provide greater coverage for unmatched recommendations or old groups.

To provide a multi-round, multi-leveled approach, we developed an incremented threshold approach. An iterative algorithm gradually increases this threshold until all original named groups have been mapped to valid recommendations or all recommendations have been mapped.

The algorithm is illustrated in pseudocode in Figure 3. Initially the change recommendation engine starts with a closeness threshold of 0, meaning that a recommendation must match an old named group exactly in both old membership and expected growth. In the above example, this means Research Group and A

would be the only match initially. For each match, if the old named group maps to a single valid recommendation the old group and recommendation are merged and added to our recommendations as a suggested evolution. This merging is done by adding the new members of the valid recommendation to the old named group.

After the merging has been completed, the old named group and recommendation are removed from the pool of possible matches because they have already been mapped. In our example groups Research Group and A would be merged and removed from the pool. Following each stage of matching and merging the threshold is incremented by 1 and the matching and merging is performed again. Thus, with the threshold now 1, Social Group is matched to B (wit$|M_{k,new}||M_{k,new}|$a closeness of 1) but not C (with a closeness of $\sqrt{3}$). Since this too is a one-to-one mapping, Social Group and B are merged and removed from the pool. Since there are no more possible old named groups in the pool, the matching and merging completes without using C.

Thus, this approach allows an increased threshold if a lower threshold will not appropriately recommend evolutions but retains the one-to-one mappings that occur more abundantly in lower thresholds.

5 Evaluation

Since this approach can be targeted at any system from which a social graph can be extracted, and since graphs can be extracted from systems with both explicit and implicit connections, we evaluated it on both types of graphs.

5.1 Explicit Graphs

We had access to two sets of explicit graphs for the purposes of evaluation. One of them was used in our previous work on foundational group recommendations in explicit graphs in Facebook (Bacon and Dewan 2011). The other was the SNAP (McAuley and Leskovec 2012) dataset. Both datasets contain information about

```
matchAndMerge(oldGroups, recommendations) {
 recommend_evolutions = empty set
 threshold = 0
 while len(oldGroups) > 0
    && len(recommendations) > 0{

  forall oldGroup in oldGroups {
   matchedVals = []
   forall recommendation in recommendations
    c = closeness(oldGroup,recommendation)
    if c <= threshold {
     matchedVals.append(recommendation)
    }
   }
   if matchedVals.size() == 1 {
    recommended_evolutions.add(
     merge(oldGroup, matchedVals[0]))
    recommendation.remove(matchedVals[0])
    oldGroups.remove(oldGroup)
   }
  }
  threshold += 1
 }
 return recommended_evolutions
}
```

Figure 3. Pseudocode for matching and merging old named groups and recommendations

multiple Facebook accounts, and both contain an unevolved social graph and a single set of unevolved ideal groups for each participant included in the dataset.

These existing data sets are a good approach for initially evaluating our composable approach, since much of past work in foundational named group recommendation was targeted at such systems. It is reasonable that a composable evolutionary approach using foundational approaches successful in such systems would also be successful.

Although these data sets contained information about ideal groups, but each dataset only had information about the social graph and ideal groups at a single moment in time. Therefore, our new model would need to somehow simulate how the graph and groups change over time. These changes in the graph and the old state of groups could be fed to the composed algorithm to generate change recommendations. These recommendations could then be evaluated against the latest version of the named groups.

We considered various alternative approaches in past work to modeling future graph growth to see if we could apply any of these approaches to our own problem domain. Previous work targeting non-collaboration domains has developed various

approaches for performing such modeling by targeting specific types of communication networks including remote sensors (Dowell & Bruno, 2001), mobile phone users (Wang, Pedreschi, Song, Giannotti, & Barabasi, 2011), and face-to-face collaboration (Scholz, Atzmueller, & Stumme, 2012). However, each of these approaches depends on physical distance between nodes, which may not be applicable to at-a-distance collaborative networks, or collaborative networks of users that are not co-located. Edges in at-a-distance graphs are not necessarily dependent on the physical distance between the nodes they are connecting. In a social graph, two nodes that are a large distance apart in reality may still have an edge connecting them (e.g. individuals on a social network that collaborate from different continents), or, similarly, two nodes that have a small physical distance may not have an edge between them. For example, past work has observed that in the social network, LiveJournal, as many as one-third of connected nodes in large networks did not share a physical locality (Liben-Nowell et al. 2005).

Other approaches have generally modeled evolution of graphs by adding both edges and vertices to a graph based on the power law (Hannon, Bennett, & Smyth, 2010; McAuley & Leskovec, 2012), which indicates that as time progresses and a social graph grows, newly added vertices in the graph are more likely to have a higher degree (or more edges to other vertices in the graph). Existence of the power law has been shown in large graph representations, such as citation networks (Hannon et al. 2010) or collaboration graphs of movie actors (McAuley and Leskovec 2012) and thus such graphs have been effectively modeled using the above technique. However, smaller graphs have been shown to be more difficult to approximate with the power-law (McAuley and Leskovec 2012).

Some social graphs, such as those mentioned above are sufficiently large such that the power-law applies or is likely to apply. However, the past work has observed that the ego-networks often are not large enough to conform to the power law (Barabási and Albert 1999; J. W. Bartel and Dewan 2013).

Because of the failure of other modeling techniques to map to our problem domain, we used a randomized approach to model past graph evolution. To model an evolution at time t_{k-1}, we assumed there is a set of new members ($M_{new,k}$) for our social graph at time t_k (S_k)., where each of these new members will be added to the social graph after time t_{k-1}. We can then subtract this new set of members from the social graph (S_k) and named groups at time t_k (G_k) to find the respective S_{k-1} and G_{k-1}.

By modeling group growth in this way, it is possible to model the growth of the social graph between time t_{k-1} and t_k and test different growth rates by adjusting the size of $M_{new,k}$. Specifically, we modeled these growth rates as a proportion of the size of global social graph by defining a *vertex growth rate* such that *vertex_grow_rate* = $|M_{k,new}|/|S_k|$ $|M_{k,new}|/|S_k|$. We varied vertex growth rate = 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.10,, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, and 0.90. Then the previous state of a group at time t_{k-1} were

modeled by removing any members of that group that existed in the graph at time t_k but not at time t_{k-1}.

By modeling according to vertex growth rate, we remove the restriction that users know when exactly to trigger evolutionary group recommendations. An application may automatically trigger evolutionary recommendations once a graph has achieved a vertex growth above some threshold. However, users may still manually trigger evolutionary group recommendations, and, assuming different approaches perform better at different vertex growth rates, the application can automatically apply the best performing approach.

Since neither data set indicated when individuals or nodes were added to the social graph, we could not deterministically choose $M_{new,k}$ to match our chosen vertex growth rates. Instead the size of each $M_{new,k}$ was chosen to match our vertex growth rate, and we randomly selected nodes in the social graph to be members of $M_{new,k}$.

The next issue is how to measure the user effort required to edit recommendations or named groups in this model. We analyzed effort in terms of cost users must exert to transform the recommended group evolution to the final ideal group. We measured this effort in the number of individuals a user must add or delete from a recommendation to match it to the ideal group. The method for modeling user effort is consistent with past work (Bacon and Dewan 2011), which model users' effort when editing members of recommended groups the number of additions and deletions users must perform, and with other user effort metrics or models that measure distance or effort based on user actions, such as Levenshtein's distance (Levenshtein 1966) or the GOMS model (Card et al. 1980). Levenshtein's distance measures the number of edits required to make two sequences equivalent, and the GOMS model, measures user effort as discrete actions such as clicks. If we divide a Levenshtein's distance into multiple dimensions, such as additions and deletions, we can then assign discrete user effort costs to each of those dimensions. For example, an addition may require a single click to specify the add operation and some expected number of keystrokes to type the name of the element, and a deletion may require a single click to delete a single element.

Moreover, it was our goal to ensure that our approach was better than manually evolving groups. Since our modeling approach is limited to the adding of individuals to graphs and groups, in the manual case users only need to add members to groups and never delete them. Therefore, to directly measure the cost of an approach when compared to manual, we evaluated both the full recommendation approach and the composed change recommendation approach using deletions (the total number of individuals that must be deleted) and relative growth additions (the percent of evolutionary additions that must be made manually).

Since the manual approach requires no deletions, if there are more than 0 deletions associated with an approach, it is worse than the manual in terms of

deletions. Since the user must perform all evolutionary additions manually in the manual case, if the relative growth additions are greater than 1, the approach is costlier than manual in terms of additions.

As this discussion above shows, users may need to delete members from recommendations even if they do not delete nodes in their social graphs. Our heuristics assume that users to do not delete nodes in the social graphs; and our

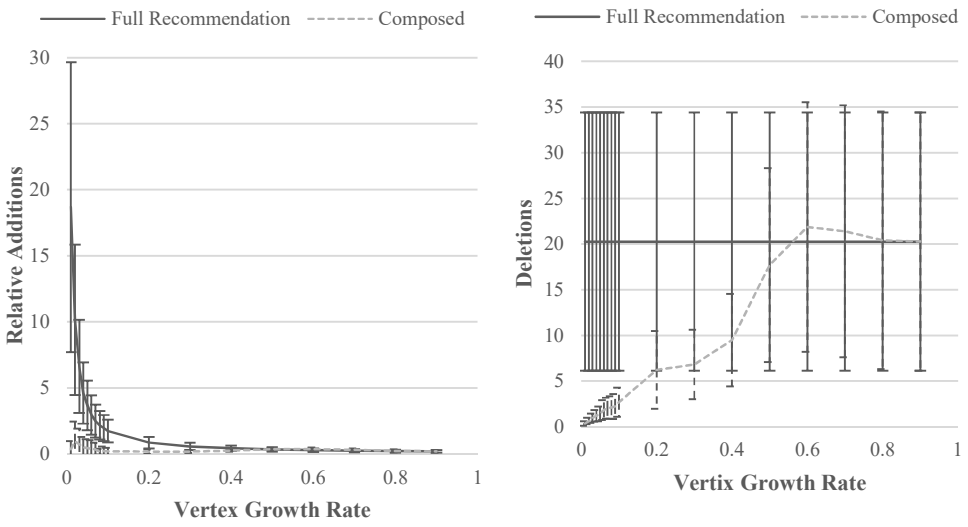


Figure 4. Effort of evolution with data from Bacon & Dewan

analysis assumes that recommendations may have spurious elements (not deleted from the social graph) that need to be deleted explicitly from the recommended set.

5.1.1 Results and Analysis

Both data sets contained multiple accounts that could be evaluated. In the case of Bacon & Dewan's data, there were 11 accounts. In the SNAP data set, there were 10 data accounts. For each account, we modeled the growth of each graph for each chosen vertex growth rate. Then we reported the mean of relative growth additions and deletions across all accounts for each data sets at each growth rate. These values for both the Bacon & Dewan data set and the SNAP data set are shown in Figure 4 and Figure 5, respectively, with error bars indicating 95% confidence intervals. Both show similar results for both relative growth additions and deletions.

In both cases, full recommendation required more than 1.0 relative additions for vertex growth rates less than 0.3 (significant with p=0.01), indicating it is costlier than manual for these low vertex growth rates. However, as the vertex growth rate increased past 0.3, this relative addition cost decreased below 1.0 (significant with p=0.01). We assumed the cost of additions is greater than that of deletions, and we accordingly ranked approaches first by additions then by deletions, we judged full

recommendation to be better than manual in all cases except when the growth rate was below 0.3. In its best case, the relative additions were less than 0.3 (coincidentally the same as our previous growth rate threshold) or a 75% reduction over the cost of additions in the manual case.

Also, in both cases, the full recommendation approach required a constant number of deletions regardless of the vertex growth rate. This is because the full

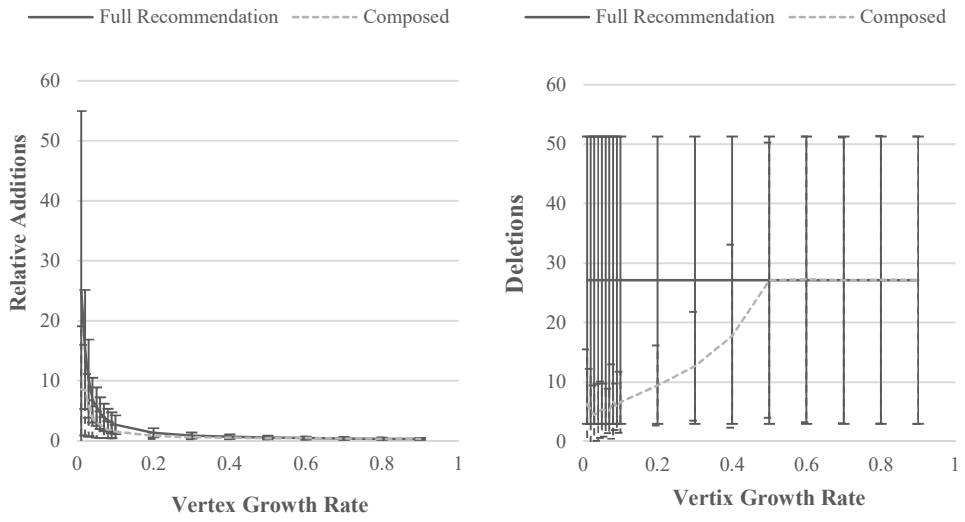
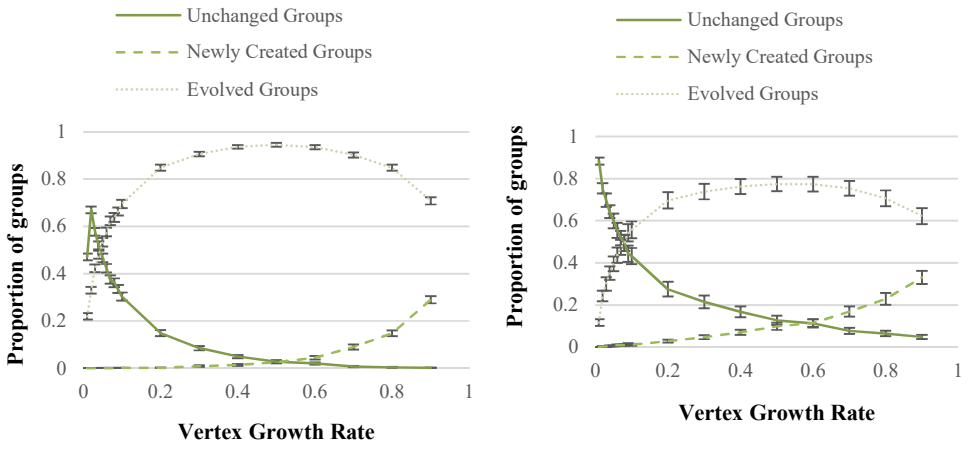


Figure 5. Effort of evolution with data from SNAP

recommendation approach always generated a whole new set of groups from the same graph, which meant it always generated the same recommendations for the same ideal groups. Therefore, these new group recommendations always required the same number of absolute deletions. Moreover, these recommendations also required the same number of absolute additions. However, because this is divided by the number of new members to compute the relative growth additions metric, the required relative growth additions for the full recommendation approach varied with different growth rates.

In terms of the composed evolutionary approach, both data sets showed a lower relative additions (deletions) value than full recommendation initially at a low growth rate. At their best cases, we found a significant improvement over full recommendation in terms of additions. In the Bacon & Dewan data set, the required relative additions were reduced by over 97%, and, in the SNAP dataset, the required relative additions were reduced by over 65%. Then, as the growth rate increases, the composed approach requires more or the same amount of additions (deletions). The point at which the composed approach costs more than the full recommendation approach varies based both on the data set used to conduct experiments and whether we are evaluating additions or deletions.

In the case of the Bacon & Dewan data set, the composed approach significantly (p=0.01) requires fewer additions or does not significantly require more additions than the full recommendation approach in terms of relative growth additions when the growth rate is greater than 0.07. The composed approach requires fewer deletions than the full recommendation approach when the growth rate remains below 0.3. At higher growth rates, we did not observe the required deletions to vary significantly between the composed and full recommendation approach with p=0.01.. As discussed previously, we assumed additions to be costlier than deletions. Therefore, we judged the composed approach to be a cheaper approach when either (a) it required fewer relative additions than full recommendation, or (b) required about the same number of relative additions and significantly fewer deletions. These criteria led us to the judgment that the composed approach is less costly than the full recommendation in the Bacon & Dewan data set when the growth rate is above 0.7. Moreover, we also judged the composed approach better than manual for these growth rates. Even though it requires a non-zero number of deletions, the number of additions is significantly lower than 1 (less than 0.2 in the best case).



(a) Data from Bacon & Dewan

(b) Data from SNAP

Figure 6. How groups change in explicit graphs

In the case of the SNAP data set, we also observed a difference in the point at which the cost of the composed approach exceeded or was equal to the cost of full recommendation. For relative growth additions, this occurred when the growth rate was ≥ 0.6, but was not significantly different with p=0.01. For deletions, this point at which the composed and foundational became equal was when the growth rate was ≥ 0.6, but again was not significantly different with p=0.01. In its worst case, the composed approach required the same mean deletions as the full recommendation approach. Therefore, we judged the composed approach to be

better than full recommendation when the growth rate was < 0.6, because it required fewer relative growth additions and the same or fewer deletions, but given the results of our t-tests, we did not judge our results in this data set to be significant, likely due to the smaller number of accounts in the data set

In the SNAP dataset, we noticed a difference in when composed outperformed manual. Instead of composed always requiring less than 1 relative growth additions, it only required less than one when the growth rate was ≥ 0.2. Based on this finding, we judged the composed approach to be better than manual only when the growth rate was greater than 0.2.

Interestingly, the point at which composed becomes costlier than the full recommendation approach may be explained in both data sets by the type of evolution that occurs in groups. As the graph grows from time t_{k-1} to time t_k, groups can evolve in three ways: (1) groups can remain unchanged (2) groups can increase from a non-zero size by adding some members that were also added to the social graph between time t_{k-1} and t_k, or (3) groups can be newly created if all their members are added to the social graph between time t_{k-1} and t_k. We measured what portion of groups fell into each of these categories, and reported them in Figure 6.

As the figure indicates, the Bacon & Dewan data showed a decreasing number of evolved groups after the growth rate exceeded 0.5, and the number of newly created groups exceeded the number of unchanged groups at the same point. Similarly, only when the growth rate exceeded 0.6 in the SNAP dataset did the number of evolved groups start decreasing and did the number of newly created groups exceed the number of unchanged groups. Both growth rate values match the thresholds at which we judged the full recommendation to perform better than the composed approach.

Intuitively, this is likely because the number of new groups is so large that it outweighs the benefits of the composed approach. Recall that the composed approach does not create new groups, only evolves existing groups, but the full recommendation approach can create new groups because it replaces all groups with a new set of groups. If a group still exists in the past, the composed approach is more beneficial, because it retains the name of the old group and only recommends updates to its members. On the other hand, if the group did not exist in the past, the composed approach will not recommend anything about that group, but the full recommendation approach will recommend groups that users must name and edit.

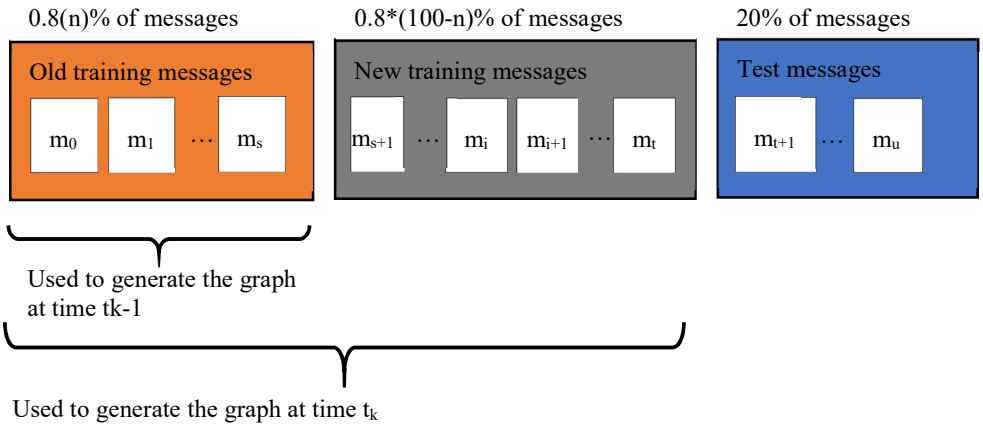


Figure 7. Division of messages for evolution experiments

5.2 Evaluation with Implicit Graphs and Message Histories

Given the success of our composed approach on explicit social graphs, we also wanted to evaluate its effectiveness on implicit graphs or graphs that were extracted from messages between users. Our motivation for doing so was based on past work that effectively mined social graphs similar to explicit ones from message histories (Fisher and Dourish 2004; MacLean et al. 2011; Roth et al. 2010).

To perform our analysis, we used anonymized email recipient from 28 participants (consisting of university students, university faculty, and public school teachers) from a separate study on email responsiveness, the 20 Newsgroups dataset ("20 Newsgroups" n.d.), and the Stack Exchange public data dump ("Stack Exchange Data Dump" 2012). Our email data collection consisted of the either the 2000 most recent messages or the 400 most recent email threads for a given user (whichever terminated first). The 20 Newsgroup dataset consists of posts of responses sampled from 20 different popular Usenet forums. The Stack Exchange public data dump consists of all questions asked on the Stack Exchange platform with a Creative Commons license.

For each of these datasets, we extracted an implicit graph by modeling nodes as their respective specified recipients in the dataset: {email addresses, Usenet groups, or Topic categories}. Intended named groups therefore contained items of the same type as nodes. An edge was formed between two nodes in these implicit graphs if the nodes co-occurred in the same email message, post, or question.

However, evaluating groups extracted from these implicit graphs could not be evaluated in the same way as groups from explicit graphs. We had only had access to the extracted groups and the nodes that co-occurred together in messages, but not any ideal groups identified by users as helpful. Therefore, it was not possible

measure how users would edit any recommended evolutions presented to them to reach ideal groups.

Instead, we needed to develop a different evaluation approach with certain goals. Namely, we needed to separate messages into three set of groups: (1) a set of older messages that are used to generate an older social graph prior to group evolutions, (2) a set of newer messages that, together with the set of older messages, would be able to generate a new social graph necessitating evolution, and (3) a set of test messages against which to evaluate our evolution recommendations. With these three groups, we could then recommend groups in the old social graph and recommend evolutions to these groups based on the new social graph, and compare the use of the old groups and their evolutions in the test messages.

To separate messages into these three groups, we developed the approach shown in Figure 7. In this experimental design, we would allocate the last 20% of all messages as the set of test messages. Then, the remaining training messages would be divided into two parts. The first portion the training messages, (called "old training messages" in the figure) would be made up of the first *n*% of the training messages. This first portion would serve as the set of older messages that would be used to generate the old social graph. The second portion (called "new training messages" in the figure) would be made up of the last (100-*n*)% of training messages. This second portion would serve as the set of new messages that would be used with the first portion to generate the new graph. The value *n* could be then adjusted to allow for different vertex growth rates of the social graph. The smaller the value of *n*, the larger the number of new training messages in relation to old ones. This larger "new training messages" is the more likely the social graph has grown by a larger amount in this period. By using older messages for training, we make use of the temporal dimension available in implicit graphs, which was not available in the data set of final explicit-graphs and thus required us to hypothesize a growth model. Here we can use actual growths for evaluation.

Using these old training messages, it was possible to generate a set of old groups using a previously used foundational recommendation approach. Recommendations for how these old groups should evolve could be generated using old graph and new graph generated from the two different portions of the training messages as described above.

We could then evaluate these evolution recommendations against the messages in the test messages. However, to perform this evaluation, we needed to identify some metrics to measure user effort to compare this new approach against a foundational approach or an approach with no recommendations.

Therefore, the next question is what metrics should be used to evaluate effort. Because there is no set of ideal groups toward which we can assume users are evolving, it is not possible to map each change recommendation to an ideal group to which the recommendations should be edited to match. This means it is not

possible to use a model where users edit recommendations as they are presented but before they are used in messages.

Instead, it is possible to assume a model that assumed users are only required to name or reject recommended groups, each of which takes effort. Future messages may then use one of these recommended groups that are not rejected if there is at least one named group where a user is not required to add all recipients in the message or remove all members of the named group. The effort for a message can then be measured as the lowest editing cost to use a group in a test message. This editing cost was reported as the percentage of a group's members that must be deleted and the number of the messages recipients that must be added manually. These values can then be averaged across messages to give mean editing costs per message.

It is possible to apply this same idea to change recommendations, where effort is measured as the how often users must reject or accept recommendations and the cost to edit groups as they are used in messages. However, the evaluation of change recommendations requires some modifications to the model used in foundational recommendations. Specifically, these modifications occur in the portion of the model where users handle new change recommendations. Change recommendations, unlike foundational ones, recommend which members should be added to an existing named group. Since these groups already have names, there is no cost in this model to name groups. Moreover, the rejection of a change recommendation does not imply that a group is not used. Instead, this rejection indicates that the group should not be changed and should remain in its previous state.

Therefore, the evaluation model and its metrics will consider these differences for change recommendations. For approaches that make change recommendations, the model will first assume the user went through the original process to name or reject groups generated from foundational recommendations based on the training messages. This effort is measured in the same way as in previous foundational experiments.

Then effort for handling change recommendations is measured as the percentage of those recommendations that are rejected. As a heuristic, we assumed users would reject recommendations that would not be useful in at least one future message. To evaluate the usefulness of our generated change recommendations, we performed the following steps:

1. We defined a concept of *candidate matching test messages* which is a message from the set test messages that fulfills two requirements: (a) Addressing the test message with the group resulting from the change recommendation does not require the user to delete all members of the evolved group or add all expected recipients in the message, and (b) the user is required to make fewer additions and deletions to transform the change recommendation to the set of recipients than to transform any existing unevolved group.

2. For each change recommendation, we identified the set of candidate matching test messages. For a poor recommendation, this set may be empty.
3. We required that no two change recommendations had overlapping sets of candidate matching test messages. If two sets contained the same test message, the test message was removed from the set corresponding to the change recommendations that required a higher cost in terms of additions and deletions. As previously discussed, we assume additions were higher cost, and thus measured first by additions and then by deletions. If the cost of two recommendations was equal, we randomly selected one.
4. Finally, we determined evolutions as useful, and therefore accepted by the user, if and only if it had a non-empty set of candidate matching test messages.

An acceptance means the old group is transformed based on the recommendation, and a rejection implies the old group is retained, but not transformed. In this way, we retain only recommended evolutions that would be better at addressing future messages than unevolved groups, and avoid overcounting any two groups that would have been accepted because they could be used to address only the same message.

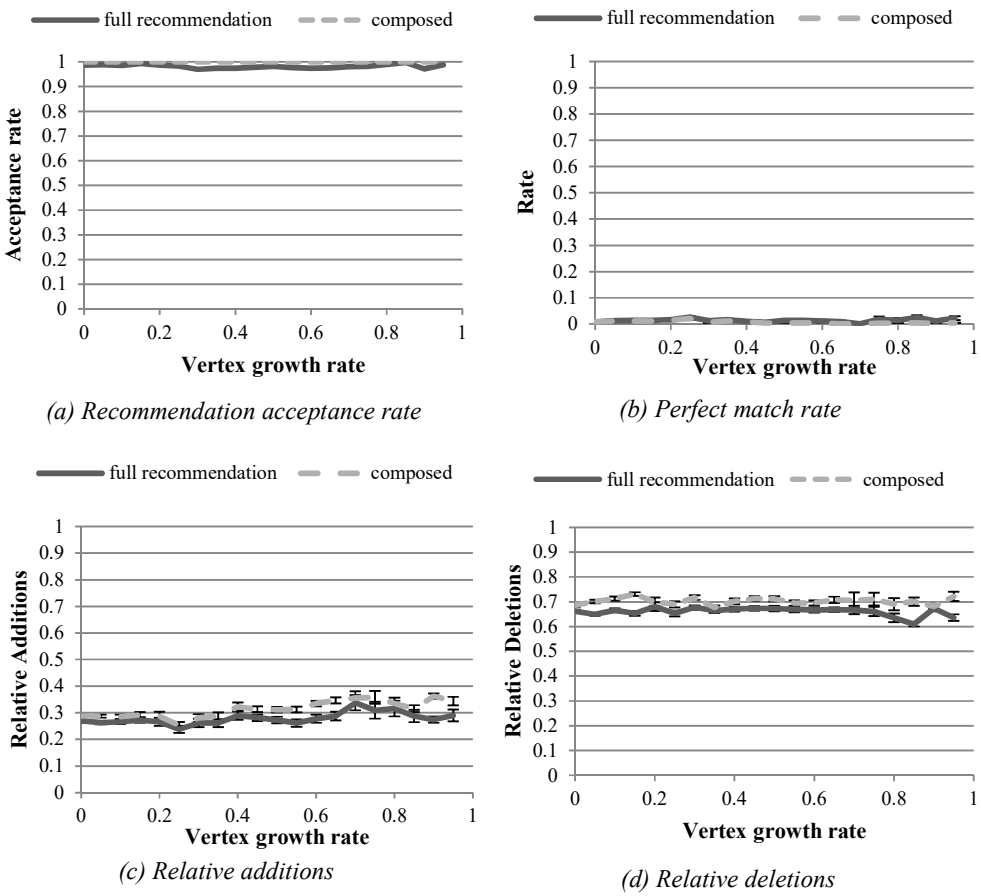
The relative additions, relative deletions, and perfect match rate metrics can then measure the effort required to address each test message using only the existing and possibly transformed groups.

To model different evolutionary rates, n was varied to achieve different rates at which the number of members in the graph increased d. Specifically, this was varied to test when the number of members increased by 1-100%. This allowed the analysis of the relative improvement as the growth rate of the graph changed.

5.2.1 Results and Analysis

5.2.2 Email

The results of evolutionary predictions in email are shown in Figure 8. As Figure 8(a) and (b) indicate, there was no effective difference in the acceptance rate or perfect match rate between the full recommendation and composed approaches to group evolutions.



(a) Recommendation acceptance rate

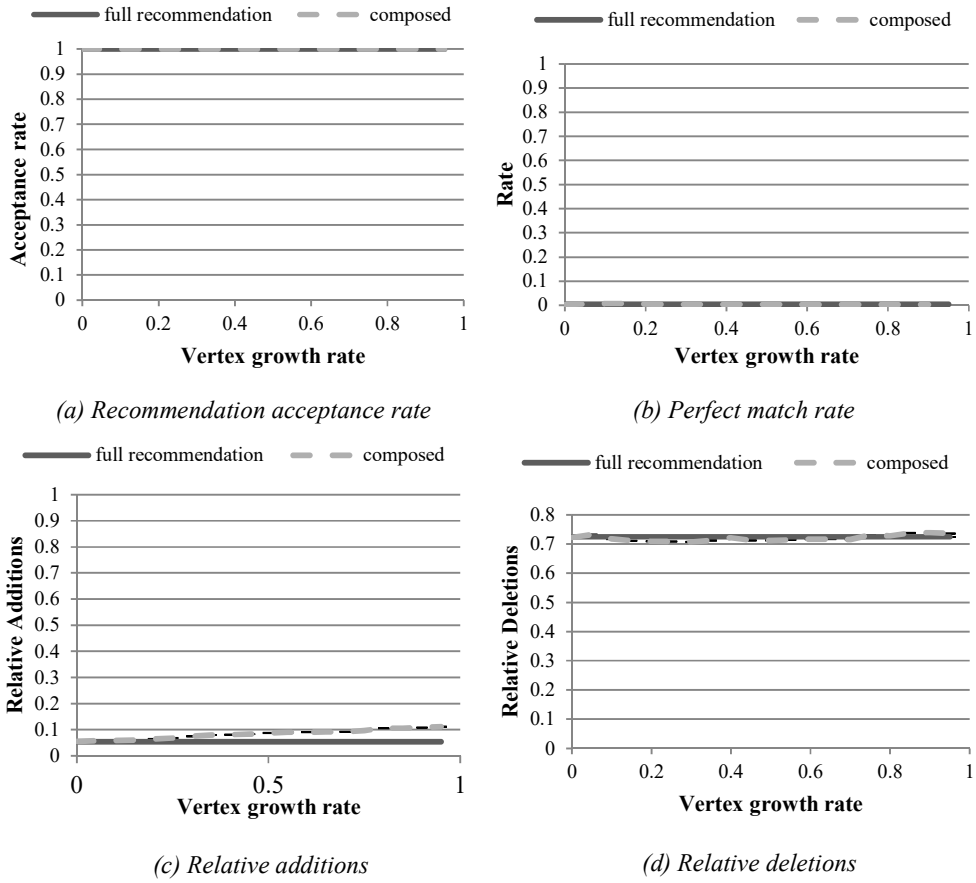
(b) Perfect match rate

(c) Relative additions

(d) Relative deletions

Figure 8. Evolution Results in email

On the other hand, both relative additions and deletions show some variation when comparing full recommendation and composed approaches. Relative additions, as shown in Figure 8(c), have little variation with smaller vertex growth rates around 0. As this growth rate increases, we observed that this variation tended to increase, with composed requiring more relative additions than full recommendations. Relative deletions did not show such systematic variation. As Figure 8(d) indicates, we observed that the composed approach always required more relative deletions than the full recommendation approach. However, in both relative additions and deletions, though we observed changes in these variations, the magnitude of these variations remained small. In fact, the variation between the results for full recommendation and compose approaches were so small that we were unable to effectively show that the values were different with $p < 0.01$.



(a) Recommendation acceptance rate

(b) Perfect match rate

(c) Relative additions

(d) Relative deletions

Figure 9. Evolution results in Usenet

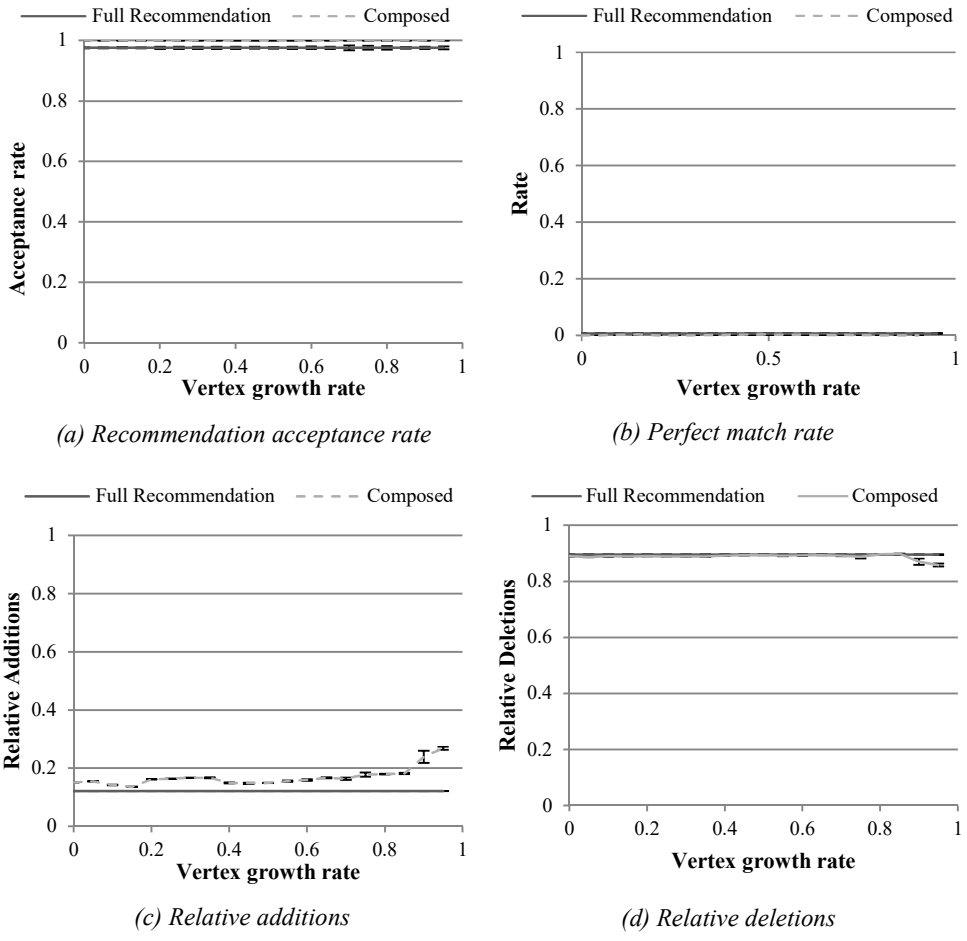
Recall that the composed approach, because it is a change recommendation approach, preserves names users assigned to groups. On the other hand, the full recommendation approach, because it replaces all groups with a new set of groups, requires that users exert effort to assign new names to all accepted groups, whether that group was previously named or not. Given that all other metrics could not be determined as significantly different, we then claim that the full recommendation requires more effort than the composed one. With all other forms of effort being equal, these two approaches appear to only differ in the cost of naming groups, which is 0 for the compose approach and non-zero for the foundational one.

Moreover, since all relative values were less than 1, we claim that that the composed approach requires less effort than manual, because it requires less effort to address future messages than if they had been addressed manually.

However, the full recommendation approach appears to always require less than 1.0 relative additions and 1.0 relative deletions. This indicates that full recommendation requires less effort than manual evolution in all cases, regardless of vertex growth rate.

5.2.3 Usenet

These results when using the 20 Newsgroups dataset are shown in Figure 9 and are like those seen in email. Both the acceptance rate and the perfect match rate were so close in the full recommendation and composed approaches that they were effectively the same (Figure 9 (a) and (b)). Relative additions also showed a systematic increase in variation between the two approaches, with the composed approach requiring more relative additions and the growth rate increased (Figure 9 (c)). Finally, some slight variation was observed in relative deletion results from the two different approaches, but this variation did not appear to show a pattern with respect to the vertex growth rate (Figure 9 (d)). As mentioned previously, such similarity between results is not particularly surprising since, as mentioned previously, Usenet and Email are likely addressed similarly, because they use the same message format.



(a) Recommendation acceptance rate

(b) Perfect match rate

(c) Relative additions

(d) Relative deletions

Figure 10. Evolution results in Stack Exchange

Unlike the email tests, the composed and full recommendation approaches yielded relative additions and deletions that were sometimes significantly different than each other's. Our results indicated that the composed approach required fewer relative deletions with $p < 0.05$ when the growth rate was between 0.10 and 0.70, inclusive. Conversely, however, the composed approach was shown to always require more relative additions than full recommendation with $p < 0.05$. Since we assume that the additions require more effort than deletions, we therefore cannot assume that the composed approach is better than full recommendation in Usenet. It is possible that the benefits of retaining of previous group names with the composed approach may outweigh the additional additions it requires. However, our metrics do not capture the cost of naming groups, and therefore we leave it to future work to make such evaluations about Usenet.

Despite this inability to show our composed approach as better than full recommendation, the results indicate that full recommendation is better than manual, just as in the email results. Full recommendation always yielded relative and deletions that were less than manual with a $p < 0.05$ and an FDR rate of 0.05. Therefore, we conclude that full recommendation was better than manual for group evolution with vertex growth rates greater than 0 and less than or equal to 1.0 in Usenet.

5.2.4 Stack Exchange

Finally, we evaluated the composed approach for predicting evolutions for named groups of tags in Stack Exchange using the Stack Overflow portion of the Stack Exchange public data dump. The results of these experiments are shown in Figure 10.

As in previous results with email or Usenet data, both the composed and full recommendation approaches yielded similar values for the acceptance rate and perfect match rate, as illustrated in Figure 10(a) and Figure 10(b). Like the Usenet results, relative deletions showed some slight decrease in the composed approach, but, again like the Usenet results, this variation showed no discernable pattern with respect to vertex growth rate as displayed in Figure 10(d). More specifically, all relative deletion values in these results fell within the range 0.885 to 0.895.

Like in Usenet, the composed approach required more relative additions than the full recommendation approach as illustrated in Figure 10(c). Moreover, t-tests confirmed this difference in costs with $p < 0.05$ and a FDR of 0.05.

As before, because we assume that additions require more effort than deletions, we cannot assume that the composed approach is better than full recommendation in Stack Overflow. As before, the benefits of retaining group names may outweigh these costs, but we have not specifically tested this tradeoff. Therefore, we cannot claim such an effect.

However, just as in email and Usenet, the full recommendation approach always yielded better results than manual in terms of relative additions and deletions with

$p < 0.05$ and an FDR of 0.05. Therefore, we judged full recommendation to be better than manual for all vertex growth rates in Stack Overflow.

6 Conclusion and Future Work

The main contribution of this work is to treat the evolution of named groups as a first-class research problem spanning a broad range of systems, tradeoffs, and approaches.

We have enumerated some of the conditions that lead to such evolution. Specifically, we have identified that named groups can evolve by adding members, removing members, or changing connections between existing members. We have also identified three broad categories of approaches for evolving groups: manual, full recommendation, and change recommendation. We have qualitatively compared them based on the inherent tradeoffs they make. Manual evolution is guaranteed to be the most accurate; full-recommendation requires no new algorithms; and change recommendation can offer the automation of full-recommendation and accuracy of manual evolution.

A change-recommendation engine can suggest evolutions to a single named group or a set of such groups, which are also associated fundamental tradeoffs. Member suggestion works incrementally by suggesting new members to a group, one by one, while a general engine describes changes to a set of existing named groups. The former can learn from the acceptance or rejection of each proposed member, while the latter can result in less user-effort.

It is difficult to develop a general change-recommendation engine as a single piece of research and existing work has made this problem tractable by focusing on specific instances of the problem, evolutionary growth. This paper has presented the first composable engine for recommending new connections to existing and new members of a social ego social-graph created from both implicit and explicit connections.

Such an engine must find the mapping between named groups recommended before and after the evolution. We have identified three concepts to perform this mapping, (i) a set of heuristics about the relationship between the growth of named groups and the underlying social graph, (ii) a metric for determining the closeness of two named groups that considers the additions and deletions required to morph one group into another, and (iii) a multi-round algorithm that gradually relaxes the closeness threshold in different rounds to find all mappings.

We have performed comparisons of this new approach with the full recommendation and manual approaches. The results of these comparisons differed significantly based on whether the social graph that was used to generate groups was explicit or implicit.

When evaluated with explicit graphs, our results show that the full recommendation approach outperforms the manual approach in explicit graph by

reducing the number of required additions and deletions in all but the smallest of social graph growths (< 0.25 vertex growth rate). We have also shown that change recommendation outperforms manual by reducing the cost of additions in most cases for two different data sets of explicit graphs. Furthermore, change recommendation outperforms full recommendation in terms of additions by over 95% in our best case.

Being a first-cut approach to meeting the stated goals, our composed approach has several drawbacks even in the case of explicit graphs, where they showed success over both manual and full recommendation approaches. The benefits drop as the relative growth of the social graph increases. This means that for high levels of growth where the number of new groups exceeds the number of unchanged groups, which corresponded to when 50-60% of the graph was made of new members in our data sets, it may still be more effective to use past approaches to generate a whole new set of recommended groups rather than recommend evolutions.

When evaluated with implicit graphs, our results show that the full recommendation approach outperforms the manual approach regardless of the growth rate of the social graph. We have also shown that our change recommendation outperforms manual by reducing the cost of additions in most cases for three different data sets of implicit graphs. However, change recommendation never outperformed full recommendation with statistical significance.

As the term “towards” in the title indicates, our composed approach has several limitations, which could be addressed by future work. We have restricted our recommendations to evolutionary growth and matches to be one-to-one mappings between past named groups and recommendations. With these restrictions, we are unable to offer recommendations in cases where a new group should be created, a named group should lose members, a single named group should evolve into multiple named groups, or where multiple named groups need to be merged into a single named group.

The heuristics used in the design of the composed approach and our evaluation of it assume that social graphs only may gain newly added nodes and never lose them. As mentioned earlier, our approach will handle deletions – but it is not optimized for them, consistent with the design principle of making the common case efficient while making the uncommon case possible. Because our models are not able to include social graph declining in size, we cannot make evaluation claims about how effective our approaches are in such cases. Future work could modify our approach to evaluate the rarer cases in which groups add members that were not newly added to the social graph or groups that lose nodes.

We propose two possible options for design of future heuristics that address this assumption: (1) Future work could modify the merging function of some old group A and new predicted group B could include members that exist in B and not A, but

also exist in the social graph at time t_{k-1}. Moreover merging could involve the removal of nodes that exist in A but not in B. (2) Future work could expand on our matching function. Instead of matching based on the Euclidean distance between two vectors, $v_{expected}$ = <0, 0, expected new members> and v_{actual} = <adds, deletes, new members>, where *expected new members* indicates the number of newly added nodes to the graph we expect to be added to the past group, matching could be based on an additional dimension to include removed members from the social, i.e. $v_{expected}$ = <0, 0, expected new members, expected removed members> and v_{actual} = <adds, deletes, new members, removed members>. In this way, future work may use approach to recommend evolution in graphs or models of graphs that include losses of nodes.

Evaluating heuristics that assume deletions presents a challenge when only static social graphs are available. One solution is to add some fictional users to the given social graph and delete them using some observed or hypothesized deletion model such as a random model.

We have also chosen a composed approach which finds foundational recommendations based on structures in the social graph and then matches those recommendations to existing groups. It may be possible to increase precision by integrating these steps.

Moreover, there are research questions implied by our work that future work may address.

Do our evolutionary predictions apply to other types of groups of users? In role-based access control systems, users are grouped into roles and access rights are assigned based on roles. If these systems evolve, it is likely that the members of roles will also evolve. It is possible that our evolutionary approaches are applicable in such systems to keep the members of roles up to date. Moreover, other work in these areas has identified methods for automatically evolving roles to a more optimal state, which includes the reassignment of rights [19]. Future work could reduce the conceptual gap between these fields to gain benefits of cross fertilization.

Would our approaches be effective for groups that serve purposes other than addressing future messages or creating ideal groups? Our evaluation of named groups did not evaluate other use cases. For example, other work has found that groups can be useful to understand social structures, specify profile information about group members, or filter incoming information (Bacon and Dewan 2011). Future work may study how well our recommendations are helpful in such cases.

Is it possible to adjust for incorrectly addressed messages? Our evaluation all messages contained the correct recipients. Past work has observed that at least 9.27% of email users have incorrectly addressed messages (Carvalho and Cohen 2008). Therefore, it is possible that some of the messages used in our work had similarly incorrect recipients. Future work can consider techniques to remove or correct messages with incorrect recipients.

Is it possible to better model the growth of explicit ego graphs? The data sets about explicit graphs contained nodes and edges in the explicit graphs, but not when edges or nodes were added. This meant that we had to model rather than replay graph growth. In practice, the addition of a node may trigger the additions of subsequent nodes, especially when the original node is highly connected. Our approach of randomly removing nodes to model growth does not respect such dependent additions. The fact that past work has successfully recommended new highly-connected members of social graphs by techniques such as clique completion seems to imply that highly connected members are frequently added late in the graph, indicating that removing highly connected nodes is not an unrealistic assumption. If future work were to create expanded data sets that marked when edges or nodes were added to explicit graphs, it may be possible to more accurately model or replay rather than model social graph growth. Another approach future work could take is to use the observed growth rate in implicit graphs (where we had the temporal dimension) to model growth in explicit graphs. Our evaluation models for the implicit graph data sets use the bursty nature of growth and longitudinal nature of the datasets by modeling growth based on order of received messages, rather than randomized samples. We did not use this approach for explicit graphs because we do not have evidence that our explicit graph data sets exhibited bursty growth like implicit graphs. Future work could explore if implicit and explicit graphs show similar growth rates.

Is it possible to better compare evolutionary effort? The cost of renaming groups was not incorporated into the analysis, which can be substantial, especially as the number of groups grows significantly. In general, it is a highly variable cost both in terms of the amount of time a user must take to identify a group name and the length of a group name they must type, and thus requires further work to model it.

What is the perceived usefulness of evolutionary recommendations? While we have used actual user data to perform our evaluations, we have not approached users to determine the perceived usefulness of our recommendations. The data were gathered by studies, performed by us and others, for purposes other than evolutionary recommendations. As a result, we were able to use a relatively large data set. It will be useful for future work on gathering such data for further evaluation of the evolution of named groups.

By addressing evolution of named groups as a first-class issue, identifying metrics and models for evaluating evolution, and presenting and evaluating a new approach designed to accommodate evolution, our work shows the potential effectiveness of evolutionary group recommendations and motivates and provides a basis for investigating these future research directions.

7 Acknowledgement

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

8 References

20 Newsgroups. (n.d.). http://kdd.ics.edu/databases/20newsgroups/20newsgroups.html

Amershi, Saleema; James Fogarty; and Daniel S. Weld. (2012). ReGroup: Interactive Machine Learning for On-Demand Group Creation in Social Networks. In *CHI '12. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, pp. 21–30.

Backstrom, Lars; Dan Huttenlocher; Jon Kleinberg; and Xiangyang Lan. (2006). Group Formation in Large Social Networks: Membership, Growth, and Evolution. In *KDD '06. Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, pp. 44–54.

Bacon, Kelli; and Prasun Dewan. (2011). Mixed-Initiative Friend-List Creation. In *ECSCW 2011: Proceedings of the 12th European Conference on Computer Supported Cooperative Work, 24-28 September 2011, Aarhus Denmark*. London: Springer, pp. 293–312.

Barabási, Albert-László; and Réka Albert. (1999). Emergence of scaling in random networks. *Science*, vol. 286, 1999, pp. 509–512.

Bartel, Jacob; and Prasun Dewan. (2012). Towards Hierarchical Email Recipient Prediction. In *8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*. Pittsburgh, PA: IEEE, pp. 50–59.

Bartel, Jacob W.; and Prasun Dewan. (2013). Evolving Friend Lists in Social Networks. In *RecSys '13. Proceedings of the 7th ACM Conference on Recommender Systems*. New York, NY, USA: ACM, pp. 435–438.

Burke, Moira; Robert Kraut; and Cameron Marlow. (2011). Social Capital on Facebook: Differentiating Uses and Users. In *CHI '11. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, pp. 571–580.

Card, Stuart K.; Thomas P. Moran; and Allen Newell. (1980). The Keystroke-level Model for User Performance Time with Interactive Systems. *Commun. ACM*, vol. 23, no. 7, July 1980, pp. 396–410.

Carvalho, Vitor R.; and William W. Cohen. (2008). Ranking Users for Intelligent Message Addressing. In *Lecture Notes in Computer Science. Advances in Information Retrieval*. Presented at the European Conference on Information Retrieval, Springer, Berlin, Heidelberg, pp. 321–333.

Daly, Elizabeth M.; Dominik Dahlem; and Daniele Quercia. (2014). The New Blocs on the Block: Using Community Forums to Foster New Neighbourhoods. In *WebSci '14. Proceedings of the 2014 ACM Conference on Web Science*. New York, NY, USA: ACM, pp. 52–61.

Fisher, Danyel; and Paul Dourish. (2004). Social and Temporal Structures in Everyday Collaboration. In *CHI '04. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, pp. 551–558.

Friggeri, Adrien; Guillaume Chelius; and Eric Fleury. (2011). Triangles to Capture Social Cohesion. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*. Boston, MA: IEEE, pp. 258–265.

Hannon, John; Mike Bennett; and Barry Smyth. (2010). Recommending Twitter Users to Follow Using Content and Collaborative Filtering Approaches. In *RecSys '10. Proceedings of the Fourth ACM Conference on Recommender Systems*. New York, NY, USA: ACM, pp. 199–206.

Kalman, Yoram M.; Lauren E. Scissors; Alastair J. Gill; and Darren Gergle. (2013). Online chronemics convey social information. *Computers in Human Behavior*, vol. 29, no. 3, May 2013, pp. 1260–1269.

Lampson, Butler W. (1974). Protection. *SIGOPS Oper. Syst. Rev.*, vol. 8, no. 1, January 1974, pp. 18–24.

Levenshtein, V. I. (1966). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, vol. 10, February 1966, pp. 707.

Liben-Nowell, David; Jasmine Novak; Ravi Kumar; Prabhakar Raghavan; and Andrew Tomkins. (2005). Geographic routing in social networks. *Proceedings of the National Academy of Sciences*, vol. 102, no. 33, August 2005, pp. 11623–11628.

MacLean, Diana; Sudheendra Hangal; Seng Keat Teh; Monica S. Lam; and Jeffrey Heer. (2011). Groups Without Tears: Mining Social Topologies from Email. In *IUI '11. Proceedings of the 16th International Conference on Intelligent User Interfaces*. New York, NY, USA: ACM, pp. 83–92.

McAuley, Julian; and Jure Leskovec. (2012). Learning to Discover Social Circles in Ego Networks. In *NIPS'12. Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. USA: Curran Associates Inc., pp. 539–547.

Miller, George A. (1956). The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review*, vol. 63, no. 2, 1956, pp. 81–97.

Molloy, Tim. (2011, November 14). Michele Bachmann Rips CBS After E-Mail Gaffe Exposes Debate Slight. *Reuters*. Accessed 26 September 2013

Newman, Mark EJ. (2008). The mathematics of networks. *The New Palgrave Encyclopedia of Economics*, 2008.

Newman, Mark W.; Debra Lauterbach; Sean A. Munson; Paul Resnick; and Margaret E. Morris. (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. In *CSCW '11. Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work*. New York, NY, USA: ACM, pp. 341–350.

Pal, Aditya; F. Maxwell Harper; and Joseph A. Konstan. (2012). Exploring Question Selection Bias to Identify Experts and Potential Experts in Community Question Answering. *ACM Transactions on Information Systems*, vol. 30, no. 2, May 2012, pp. 10:1–10:28.

Palla, Gergely; Imre Der´enyi; Ill´es Farkas; and Tam´as Vicsek. (2005). Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, vol. 435, 2005, pp. 814–818.

Peña, Jorge; and Nicholas Brody. (2014). Intentions to hide and unfriend Facebook connections based on perceptions of sender attractiveness and status updates. *Computers in Human Behavior*, vol. 31, February 2014, pp. 143–150.

Reeder, Robert W.; Lujo Bauer; Lorrie Faith Cranor; Michael K. Reiter; Kelli Bacon; Keisha How; and Heather Strong. (2008). Expandable Grids for Visualizing and Authoring Computer Security Policies. In *CHI '08*. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, pp. 1473–1482.

Roth, Maayan; Assaf Ben-David; David Deutscher; Guy Flysher; Ilan Horn; Ari Leichtberg; et al. (2010). Suggesting Friends Using the Implicit Social Graph. In *KDD '10*. *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, pp. 233–242.

Shen, HongHai; and Prasun Dewan. (1992). Access Control for Collaborative Environments. In *CSCW '92*. *Proceedings of the 1992 ACM Conference on Computer-supported Cooperative Work*. New York, NY, USA: ACM, pp. 51–58.

Soghoian, Chris. (2008, January 23). Exclusive: The next Facebook privacy scandal. *CNET*. http://news.cnet.com/8301-13739_3-9854409-46.html. Accessed 14 November 2013

Stack Exchange Data Dump. (2012, August). *ClearBits*. http://www.clearbits.net/creators/146-stack-exchange-data-dump

Vaidya, Jaideep; Vijayalakshmi Atluri; Qi Guo; and Nabil Adam. (2008). Migrating to Optimal RBAC with Minimal Perturbation. In *SACMAT '08*. *Proceedings of the 13th ACM*

Symposium on Access Control Models and Technologies. New York, NY, USA: ACM, pp. 11–20.