Evolving Friend Lists in Social Networks

Jacob Bartel, Prasun Dewan
University of North Carolina
Chapel Hill, NC
{bartel,dewan}@cs.unc.edu

ABSTRACT

In a social network, users can sort members of their social graph into friend lists to both understand the social structures within the graph and control the flow of incoming and outgoing information. To reduce the user-effort required to create these lists, previous work has developed techniques for generating friend-lists in a static social graph. This paper considers the user effort required to create friend lists in an evolving graph. We have developed several new initial quantitative metrics to capture this effort, and identified an initial technique for modeling graph growth. We have used these metrics and model to compare two techniques for evolving friend lists when the social graph grows: manual evolution – the user evolves friend lists using no external tools – and full recommendation – an existing state of the art tool recommends a whole new set of friend lists. In these comparisons, we used the friend lists of 12 individuals, and simulated the growth of their social graphs and friend lists using our graph-growth model. Intuitively, when the graph evolves by a small (large) amount, the manual (automatic) approach should perform better. Our experiments show that full recommendation performs better than manual when the social graph changes by more than 1%, and yields an almost complete reduction in effort in the best cases.

Categories and Subject Descriptors

H.5.3 [**Information Systems**] Group and Organization Interfaces – *computer-supported cooperative work, evaluation/methodology*

General Terms

Algorithms, Management, Measurement, Experimentation, Security, Human Factors.

Keywords

Evolution; social networks; Facebook; friend lists; refactoring.

1. INTRODUCTION

Currently, social network users have a large number of friends with whom they share large amounts of information in a variety of forms, such as posts, messages, photos, and links. One recent statistic stated that the average Facebook user has 130 friends and shares 90 pieces of content with those friends per month [8]. With such broad and varied forms of communication, certain tasks become difficult. A user must be able to organize their friends to understand the social structures to which they belong, filter out and sort incoming information, and control where outgoing information is directed.

One approach to reduce this difficulty is to group users sharing some characteristic(s) and apply the same information controls to each member of a group., Several popular social networks today allow an arbitrary number of friend lists as they are called in Facebook (and tags, groups, or circles in Renren, LinkedIn, and Google+). Previous studies have found that users either do not know how to create these lists [13] or are not willing to put the time and effort required to create them [3, 8, 9, 13]. In one health study [10], people who wished to share their medical problems with others did not use Facebook to do so because of the fear that this information may reach unintended ears. This is consistent with a more general problem of the high effort cost in access control lists [10],

The question then becomes how to reduce the costs of creating these friend lists. One approach is to create a recommendation tool to generate these lists [2, 3, 6, 9]. With such a tool, the user only has to approve, disapprove, or modify a recommended list rather than learn how to create a list and then generate a list from scratch, leading to significantly less learning and effort costs. Previous research has shown that the cost of manually editing recommended lists is a small fraction of the effort required to manually create the lists from scratch. However, current friend list recommendation tools make recommendations only when a friend list is initially created [3, 6, 12], or are limited to recommendations for two lists [7]. How this arbitrary number of lists should evolve as the social graph changes has been explicitly stated as an open problem by authors of these tools [2, 3, 9].

This paper takes a first stab at addressing this problem. It identifies initial new quantitative metrics to capture the effort a user has to exert to manually edit, accept, or reject recommendations In addition, it derives an initial technique for modeling graph growth It uses these metrics and model to compare two approaches for friend list evolution: manual evolution - the user evolves (manually or automatically generated) friend lists using no tools and full recommendation – a state of the art friend list recommendation tool is used to recommend a whole new set of friend lists.

2. FRIEND LIST RECOMMENDATION DESIGN SPACE

Our problem is a sub-problem of the general problem of finding communities in a network of nodes [3]. Our focus is on recommending classification of friends of a particular user into (potentially overlapping) friend lists. There have been a variety of diverse approaches to this problem in previous work. To combine the ideas in these approaches and to better understand and describe our own work, we have developed a design space of friend list recommendations, which is a contribution in its own right. These approaches can be divided into two broad schemes, member suggestion [2, 12] and group creation [3, 6, 8, 9] .

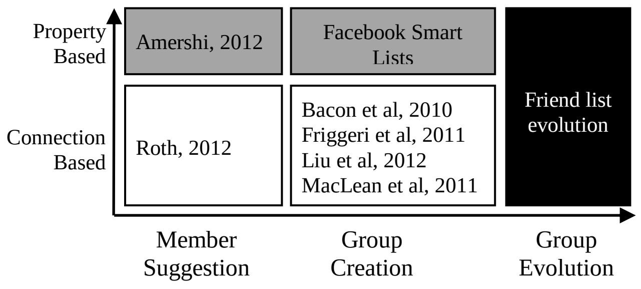

Figure 2. Design space of friend list recommendations

Member suggestion recommends friends who should be added to a new friend list. A user initially has an unpublished empty friend list, for which a recommender suggests new members. From those suggestions, the user selects appropriate additions to the friend list. The user then may accept this list or request more recommendations, forming a cycle in the work flow where each iteration of the cycle requires user input. This process is then carried out for other friend lists.

In group creation, the user starts with no friend lists. The recommender suggests a set of new friend lists by determining clusters within the user's set of friends, each of which the user can edit or reject. The edited friend lists are then published. As opposed to the incremental member suggestion approach, which may require user input multiple times for recommendations for each friend list, the "batch" group creation approach requires user input at only one point for all recommendations.

Recommendation schemes can be further categorized based on how they group users. Again, there are two main approaches. The first is property-based clustering, which clusters individuals based on shared values of associated attributes such as age, sex, or location. The second is connection-based, which clusters individuals based on subsets of vertices in the social graph that are highly connected. Each of these approaches has its strengths and weaknesses. The property-based approach uses more information, and thus, is likely give predictions that are at least as precise as those of the connection-based approach. On the other hand, because it is dependent on the individuals' properties, it is restricted to systems and users where these properties are available and correct. Some systems may grant limited access to these properties due to privacy or security reasons, and many users may specify an incorrect or incomplete set of properties, thus limiting this technique to subset of social networks.

These different techniques can be sorted into the design space, shown in Figure 1. As indicated, there has been work in all points involving member suggestion and group creation in this space.

3. FRIEND LIST EVOLUTION

Previous approaches offer recommendations only at a friend list's creation time. However, social networks can gain and lose members and connections, which would require the evolution of friend lists. Such evolution is not restricted to social networks. It occurs in any system that can be improved or expanded while retaining some of its old functionalities, and is often tool-supported. For example, database refactoring systems restructure databases as they grow to improve the architecture while retaining the original information, and code refactoring systems update code to improve organization and readability while retaining the original behavior. Our contribution is to consider this issue in the context of friend lists.

There are three ways that a social graph can change: members can be added, members can be removed, and connections can be changed. In this first cut at friend list evolution, we address only adding individuals and connections to a user's social graph and friend lists. Thus, our work applies directly to only those users who do not do "garbage" collection to delete connections or regroup users in a social graph.

To illustrate this evolution, we introduce a running example illustrated in Figure 2. In the past (time t_{k-1}) Joe's social graph contained a 6 friends, Alice, Bob, Carol, David, Eva, and Frank, and two friend lists, Research Group and Social Group (Figure 2 (a)). Between time t_{k-1} and $t_{k,}$ Joe's company hired two people, Greg and Hal, and Joe's social graph grew to include them (Figure 2(b)). Joe also wants each of his friend lists to grow to include his new friends (Figure 2 (c)).

One way Joe could reach these ideal lists is the manual approach, where he determines what has changed in the social graph between time t_{k-1} and t_k and evolve the friend lists accordingly. This approach has the possibility of being the most precise because it takes all of a user's intentions into account. However, it requires significant work if the number of friends or friend lists is significantly large or there are significant changes to the social graph. Moreover, this approach requires that the user identify changes between the different states of the social graph, which may not be feasible if the social graph is significantly large or has changed a significant amount.

Full recommendation uses a friend list recommendation tool, or a recommender engine, to generate a new set of friend lists from the evolved graph, which the user then edits and labels. As mentioned earlier, friend list recommendation can be divided into member suggestion and group creation. Each of these approaches has its pros and cons in terms of group evolution. To illustrate, we

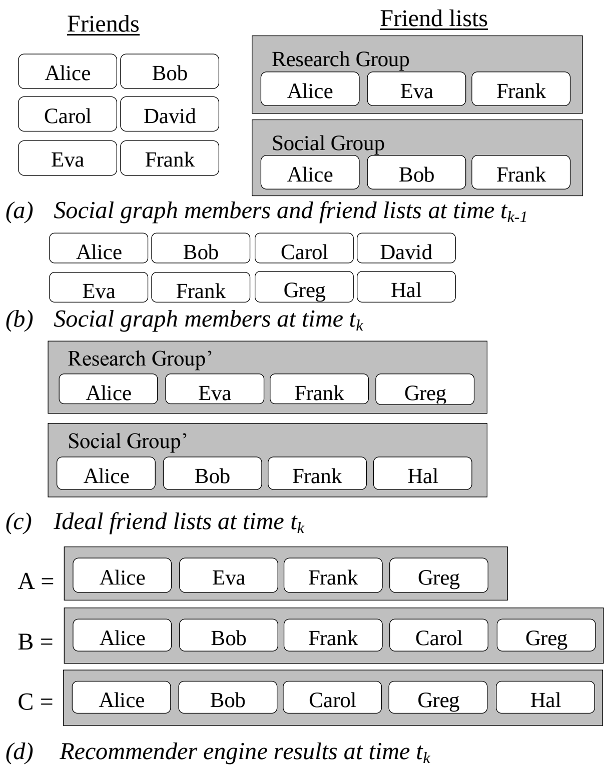

Figure 1. Example of friend list growth

continue with our running example where the social a recommender engine generates the new friend lists in Figure 2(d).

Consider the scenario where Greg and Hal are added to `Social Group` and `Research Group` remains unchanged. In general, member suggestion, the user may first unnecessarily ask for recommendations for `Research Group`. Alternatively, group creation may only present the user recommended changes to `Social Group`. If the recommendations made using member suggestion and group creation are both perfectly effective, then it can be assumed that in group creation a user will only handle friend lists that change, leading to possibly fewer steps.

However, group creation cannot use feedback to adjust recommendations, while member suggestion's iterative nature allows feedback throughout the recommendation process, which can be used to improve future recommendations. Consider the situation in which Greg and Hal should be added to `Research Group` and `Social Group`, respectively. Let us assume that without user feedback, both approaches predict the incorrect list for each user. After the user overrides its first wrong suggestion, member suggestion can correct its second prediction. Group prediction would require the user to correct both predictions.

In this paper, we focus on group creation because it has been more extensively covered by past work (including our own). We present the intuitive analysis above to show that it would be useful to also look at member suggestion in future work.

Thus, in our comparison of the two evolution approaches, manual and full recommendation, we assume that the recommender engine uses group creation. Assuming such an engine is effective, which has been shown to be true in past work by ourselves and others, full recommendation could work better than the manual approach. However, the recommender engine 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 friend list recommender.

4. EVALUATION AND ANALYSIS

In order to test our scheme, we analyzed a variety of methods for evaluating our work. The first option we investigated was to generate artificial networks. However, such an option may not match reality. The most realistic option, of course, is to use a field study observing how users actually evolve their friend lists. However, such a study would possibly take years to complete and, more importantly, due to the limited control it allows, it would be unable to compare different approaches using the same users. A third option would be to perform a lab study at a single point of time. Such an approach, however, requires us to model friend list and social graph growth.

This is the approach we took. We used data from a previous study in which 12 users were asked to create friend lists from their current social graph [3], and generated previous evolutions of friend lists and social graphs. There has been a large amount of work on modeling future graph growth, but little done to model how a graph may have grown to its current state.

We looked at these various alternative approaches to modeling future graph growth to see if any are applicable to our own problem domain. One such approach, which is used for modeling remote sensor networks, randomly distributes nodes on a plane or

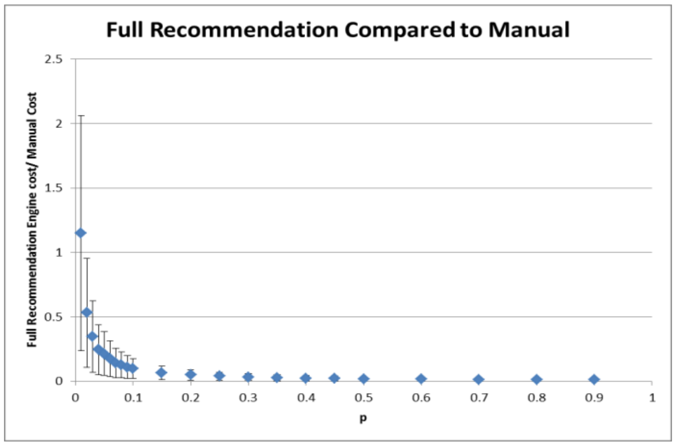

Figure 3. Cost of full recommendation compared to manual

three-dimensional surface. Then based on the how close two nodes are to each other, an edge may be formed between the two nodes [5]. This approach is not compatible with our domain as edges are not dependent on the closeness of the nodes they are connecting. In a social graph, two nodes that are a large distance apart when plotted geometrically may still have an edge connecting them, or, similarly, two nodes that have a small geometric distance may not have an edge between them.

There has also been work modeling graph growth based on the power law, which states that the number of nodes with degree d is proportional to the value $1/d^{\alpha}$ where $\alpha \geq 0$. Using this relationship, when a vertex is added to a graph at time t, it will connect to some existing vertex with probability $d_{v,t}/2e_t$ where $d_{v,t}$ is the degree of vertex v at time t and e_t is the number of edges at time t [1, 4]. Existence of the power law has been shown in large graph representations, such as citation networks [11] or collaboration graphs of movie actors [4] 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 [4], and since our graphs are only one-hop sub-graphs of the global social graph, our graphs are not large enough to be appropriately modeled using the power-law. Moreover, upon analyzing our user study data, the social graphs for our users did not display an adherence to the power law.

Because of the failure of these 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 would have been 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 friend lists at time t_k (F_k) to find the respective S_{k-1} and $F_{k-1.}$

With this model of group growth, we can model the growth of both the social graph and friend lists between time t_{k-1} and t_k and vary 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 proportion $p = |M_{new,k}|/|S_k|$ and varied p over a wide range of values as shown in Figure 3's x-axis.

This approach is limited in that it cannot model member deletion. Because we only have friend lists and social graphs from a single time slice, our data does not give us any information of individuals that are currently missing from the S_k but were contained in some previous S_j where $t_j < t_k$. For this reason, we restricted ourselves to only growing social graph and friend lists.

The next issue is how to measure the user effort required to grow friend lists generated using this model. One alternative is to ask each user to actually use each of the compared approaches. This is the approach used, for instance, in [9], where users were asked to create initial friend lists either manually or by editing generated friend lists. The experimenters found that the manual task was so tedious that few subjects completed it. This problem would be exacerbated in work that addresses evolution of friend lists, because each user would be required to evolve a friend list by different degrees. Therefore, we used an analytical approach to measuring user effort.

When evolving friend lists or using recommended evolutions, the user can exert effort by adding or removing members of a friend list or recommendation. Therefore we approximated the cost of an approach as the total number of additions and deletions that would be required to transform the old or recommended friend lists to the ideal friend lists. We then compared approaches by comparing their total costs. If there are huge differences in the costs of two approaches, then we assume that a user study would show that users prefer the approach with the lower cost.

This metric also captures the importance of the classical metrics precision, or the number of correct recommendations, and recall, or the number of times that recommendations are generated. If there is low precision, many of the recommendations will be incorrect and therefore the user will need to perform many additions and deletions during editing. Moreover, if there is low recall, there will be many failed recommendations, and thus there will be many friend lists which will require manual evolutions, which will likely require many more additions and deletions. Therefore, if user effort is shown to be low in terms of the number of required additions and deletions, we assume the precision and recall will be sufficiently high such that the user does not become frustrated to the point to stop using the recommender system.

Using the techniques for modeling evolution and user effort, we compared the full recommendation and manual approaches, which is shown in Figure 3. In this figure we plotted the average relative costs with bars marking values one standard deviation away. As shown, full recommendation performs better than manual in all cases except when the graph grew by 1%. This aligns with the results of previous work which showed the effectiveness of a recommender recommending friend list creation [2, 3, 9]. What is surprising is that such a recommender is also effective when the graph grows by a very small amount.

5. CONCLUSION AND FUTURE WORK

This paper has made multiple contributions. First of all, we have developed a design space of different approaches for recommending friend lists in social networks. This design space allows an easier comparison of past approaches and determination of which areas may be explored in future work.

We have also identified how recommender engines may be applied to evolve friend lists. To analyze this approach, we have identified a method for modeling social graph and friend list growth and identified metrics applicable towards gauging user effort. Our results show that the full recommendation approach outperforms the manual approach by reducing required additions and deletions in all but the smallest of social graph growths.

Our results are not limited to Facebook. To apply our techniques, a social network needs to be able to represent a user and his friends in a graph, with vertices representing individuals in the network, and edges representing connections between them. This social graph also needs to be able to evolve by at least allowing a user to add friends and connections to his social graph. Furthermore, the user must also be allowed to sort any of their friends into non-static friend lists. Renren, LinkedIn, and Google+ are additional systems fulfilling these requirements.

Of course, our results have limitations. We have restricted our recommendations to friend list growth. It is important to model deletions, which can make certain friend lists disappear. In addition, full recommendation forces users to rename their friend lists for each evolution. We have not modeled the cost of such renaming, which may be significant in some cases. It may also be possible to predict how friend lists should change rather than predict a new set of friend lists, which may allow such a system to automatically associate names with the recommended friend list evolutions.

We have only addressed evolutions in terms of the membership of friend lists. It may be possible to evolve access rights as the members evolve. Future work could address how to evolve rights alongside the social graph and friend lists. Our work provides a basis for investigating these unresolved issues.

6. ACKNOWLEDGEMENT

This research was supported in part by NSF grant IIS-0810861. We thank the reviewers for their in-depth comments.

7. REFERENCES

[1] Aiello, W. et al. 2001. Random Evolution in Massive Graphs. *Proc. FOCS*. (2001).

[2] Amershi, S. et al. 2012. ReGroup: Interactive Machine Learning for On-Demand Group Creation in Social Networks. *Proc. CHI* (2012).

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

[4] Barabási, A.-L. and Albert, R. 1999. Emergence of scaling in random networks. *Science*. 286, (1999), 509–512.

[5] Dowell, L.J. and Bruno, M.L. 2001. Connectivity of random graphs and mobile networks: validation of Monte Carlo simulation results. *Proc. SAC* (2001).

[6] Friggeri, A., G. Chelius and Fleury, E. 2011. Triangles to Capture Social Cohesion. *The Third IEEE International Conference on Social Computing* (2011).

[7] Hannon, J. et al. 2010. Recommending twitter users to follow using content and collaborative filtering approaches. *Proc. RecSys* (2010).

[8] Liu, Y. et al. 2012. Simplifying Friendlist Management. *Proc. WWW* (2012).

[9] MacLean, D. et al. 2011. Groups without tears: mining social topologies from email. *Proc. IUI* (2011).

[10] Newman, M.W. et al. 2011. "'It's not that I don't have problems, I'm just not putting them on Facebook': Challenges and Opportunities in Using Online Social Networks for Health". *Proc. CSCW* (2011).

[11] Redner, S. 2004. *Citation statistics from more than a century of physical review*. Technical Report #0407137. physics.

[12] Roth, M. et al. 2010. Suggesting Friends Using the Implicit Social Graph. *Proc. KDD* (2010).

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