Simplifying Friendlist Management

Yabing Liu
Northeastern University
ybliu@ccs.neu.edu

Bimal Viswanath
MPI-SWS
bviswana@mpi-sws.org

Mainack Mondal
MPI-SWS
mainack@mpi-sws.org

Krishna Gummadi
MPI-SWS
gummadi@mpi-sws.org

Alan Mislove
Northeastern University
amislove@ccs.neu.edu

ABSTRACT

Online social networks like Facebook allow users to connect, communicate, and share content. The popularity of these services has lead to an *information overload* for their users; the task of simply keeping track of different interactions has become daunting. To reduce this burden, sites like Facebook allows the user to group friends into specific lists, known as *friendlists*, aggregating the interactions and content from all friends in each friendlist. While this approach greatly reduces the burden on the user, it still forces the user to create and populate the friendlists themselves and, worse, makes the user responsible for maintaining the membership of their friendlists over time.

We show that friendlists often have a strong correspondence to the structure of the social network, implying that friendlists may be automatically inferred by leveraging the social network structure. We present a demonstration of Friendlist Manager, a Facebook application that proposes friendlists to the user based on the structure of their local social network, allows the user to tweak the proposed friendlists, and then automatically creates the friendlists for the user.

1. INTRODUCTION

Online social networking sites (OSNs) like Facebook are now a popular way for users to connect, communicate, and share content. As a result, OSNs have changed the role of users on the Web: Today, instead of just being content *consumers*, individual end users are now required to be both content *consumers* and *managers*. In other words, for every piece of information a user uploads to Facebook—every wall post, photo, status update, and video—she must decide which other users should be able to access the content. With an average of over 130 friends per user [3] and 90 pieces of content uploaded per user per month [3], the result is a significant burden both in terms of *aggregating content from friends* and *setting access control for one's own content.*

To help users share content with members of their social network, Facebook introduced the *friendlist* mechanism [4]. Each friendlist contains a subset of a user's Facebook friends, and the membership of the friendlist is private to the user by default. Facebook then allows users to (a) share content only with members of the friendlist and (b) aggregate and view the content that is uploaded by the members of the friendlist. For example, a user can create a friendlist containing all her high school friends whom she is friends with on Facebook. She can then share content with only her high school friends, and also selectively view all the content that is uploaded by her high school friends.

WWW 2012 Companion, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1230-1/12/04.

Unfortunately, the usefulness of the friendlist mechanism is impeded by two large burdens that are placed on the user. First, there are no friendlists created by default; the user must have the foresight to determine the appropriate set of friendlists to create.[1] Second, the user alone is responsible for populating the membership of these friendlists, and maintaining the correct membership over time as relationships are formed and changed. As a result, it is unsurprising that many Facebook users do not use the friendlist feature [7].

In this paper, we take the first steps towards addressing this situation by automating the creation and maintenance of Facebook friendlists. Recent work [7] has shown that the friendlists have a correspondence with the structure of the social network. In particular, many of the user-created friendlists correspond to communities in the user's 1-hop social network, and hold the potential to be detected by existing community detection algorithms (e.g., [1]). We present the Facebook application *Friendlist Manager*, which analyzes a user's 1-hop social network, detects communities, proposes the communities to the user as friendlists, allows the user to tweak the friendlists, and finally creates the friendlists for the user. Additionally, Friendlist Manager examines any previously existing friendlists to locate friends who may be tightly connected to friends in the friendlist but not a member of the friendlist; if such friends are found, they are also presented to the user for consideration.

In the remainder of this paper, we discuss motivation and implementation of Friendlist Manager in Section 2. We then present the user interface of Friendlist Manager in Section 3, and describe a small experimental deployment.

2. AUTOMATICALLY INFERRING LISTS

The intuition behind Friendlist Manager is that users typically group their friends based on shared attributes (e.g., friends who attended the same high school, friends who work for the same employer, members of the same family,

[1]Recently, Facebook introduced *smart lists* [5], which are essentially automatically created friendlists based on profile attributes (such as high school attended). Unfortunately, many users do not provide the detailed attribute information necessary, so the *smart lists* created by Facebook typically only contain a subset of the "correct" users.

Network	Nodes	Links	Avg. deg.
Facebook City A [14]	3.0 M	46 M	15.2
Facebook City B [14]	2.9 M	40 M	14.2
Facebook New Orleans [13]	63 K	1.6 M	25.6

Table 1: Number of nodes, links, and average user degree in the samples of the Facebook networks used to evaluate community detection algorithms.

or friends who have similar interests). Our intuition is supported by previous studies [8] that demonstrate the existence of attribute-based communities on Facebook. In this work, we leverage this observation to provide the users with more natural and effective ways to aggregate friends' content and express privacy controls on their own content.

2.1 Detecting communities

Community detection in large networks is a well-studied problem with a rich literature and many proposed approaches [1,6,9,10,12]. We leverage existing work on community detection to automatically infer friendlists. We require two types of community detection algorithms: one, which detects remaining members of a community if a partial set of community members are given, and second, those which globally partitions a given network into multiple communities. The use of above two classes of algorithms helps us to to expand existing friendlists (i.e those which the user has already created) and to create new friendlists for the user. It should be noted that new friendlists created can be overlapping friendlists, because a friend of a user could belong to multiple friendlists (e.g. a friend could be part of a "football team" and also a part of "hometown friends").

Since Facebook only allows applications to have knowledge of each user's 1-hop social network (i.e., only the user's friends and the connections between them), we use the 1-hop network of a user to detect communities. To create new friendlists for a user, we use the BGLL algorithm proposed by Blondel et al. [1].[2] In brief, BGLL works in the following way: It initializes each node of the graph as an isolated community. Then, it searches for ways to merge two existing communities that would result in the highest increase in modularity [9]. The algorithm continues until all of the nodes are in a single community, and then picks the stage in the algorithm with the highest modularity as the final division. BGLL has a very low computational overhead, allowing our application to give results quickly.

Unfortunately, the BGLL algorithm only assigns each friend to a single community. To further detect overlapping friendlists for the user, we expand each of the BGLL-detected communities using the algorithm proposed by Mislove et. al. [8] that detects other members of a community, given a partial set of community members by finding members (outside the existing community) which are tightly connected to the existing members.

Finally, to propose new members to existing friendlists, we again leverage the algorithm by Mislove et. al. to expand existing communities [8].

[2]We experimented with other off-the-shelf community detection algorithms as well and observed similar results. More details are in Section 2.2.

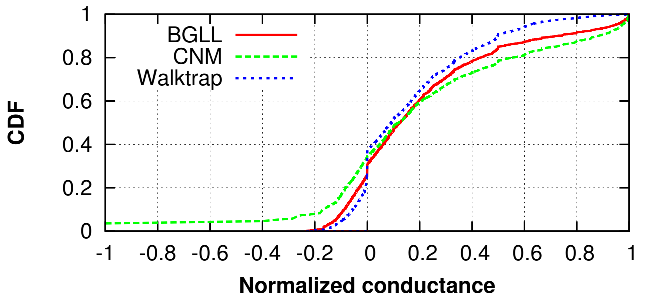


Figure 1: Cumulative distribution of the maximum normalized conductance values per user in Facebook New Orleans [13]. In 40% of cases BGLL creates communities with conductance greater than 0.2, and all methods create communities with almost equally good conductance values(the results are similar for the other two Facebook networks).

2.2 Simulating Friendlist Manager

In this section, we evaluate the potential for detecting communities in the 1-hop network of a user using community detection algorithms. We focus only on detecting non-overlapping communities in the user's 1-hop network. Work in [8] shows evidence of existence of overlapping communities in social networks and evaluates the possibility of expanding the set of potential community members given a seed set of existing members.

In order to rate the "quality" of a detected community, we use normalized conductance (a metric proposed in [8]) to evaluate the community's strength. Normalized conductance is defined as the fraction of the community's links that lie within the community, relative to what would be expected from a random graph with the same degree distribution. Similar to modularity, the value of normalized conductance varies from -1 to 1, with a positive value indicating community quality better than random, 0 representing no more community structure than random and a negative value indicates worse community quality than a random graph.

We use 3 datasets collected from Facebook to evaluate the quality of communities detected. The first two are Facebook City A and Facebook City B, two (anonymized) Facebook regional networks collected by Wilson et al. [14]. We also

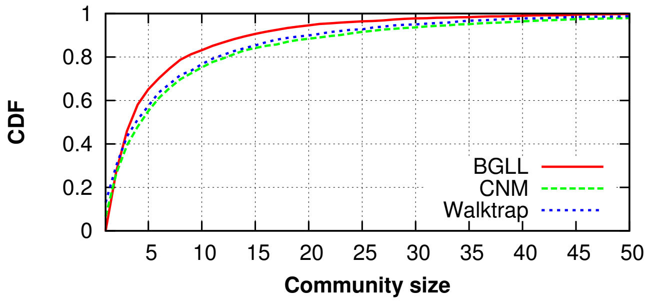
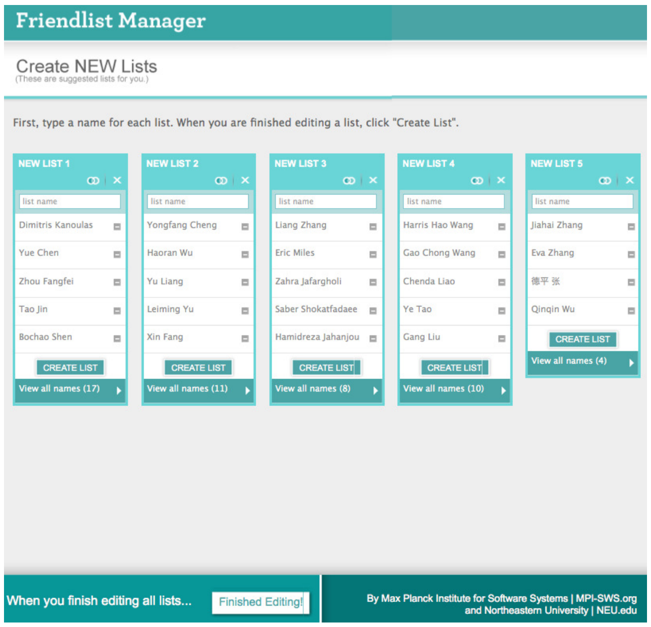


Figure 2: Cumulative distribution of the size of the communities per user in Facebook New Orleans. In more than 90% of cases, BGLL creates communities with fewer than 20 members.



Friendlist Manager

Create NEW Lists
(These are suggested lists for you.)

First, type a name for each list. When you are finished editing a list, click "Create List".

NEW LIST 1	NEW LIST 2	NEW LIST 3	NEW LIST 4	NEW LIST 5
list name	list name	list name	list name	list name
Dimitris Kanoulas	Yongfang Cheng	Liang Zhang	Harris Hao Wang	Jiahai Zhang
Yue Chen	Haoran Wu	Eric Miles	Gao Chong Wang	Eva Zhang
Zhou Fangfei	Yu Liang	Zahra Jafargholi	Chenda Liao	德平 张
Tao Jin	Leiming Yu	Saber Shokatfadaee	Ye Tao	Qinqin Wu
Bochao Shen	Xin Fang	Hamidreza Jahanjou	Gang Liu	CREATE LIST
CREATE LIST	CREATE LIST	CREATE LIST	CREATE LIST	View all names (4)
View all names (17)	View all names (11)	View all names (8)	View all names (10)	

When you finish editing all lists... Finished Editing!

By Max Planck Institute for Software Systems | MPI-SWS.org
and Northeastern University | NEU.edu

Figure 3: Screenshot of user interface of the application with proposed new friendlists.

use the Facebook New Orleans network [13]. The high-level statistics of these networks are shown in Table 1. We selected 1,000 users at random from each of these networks for our study.

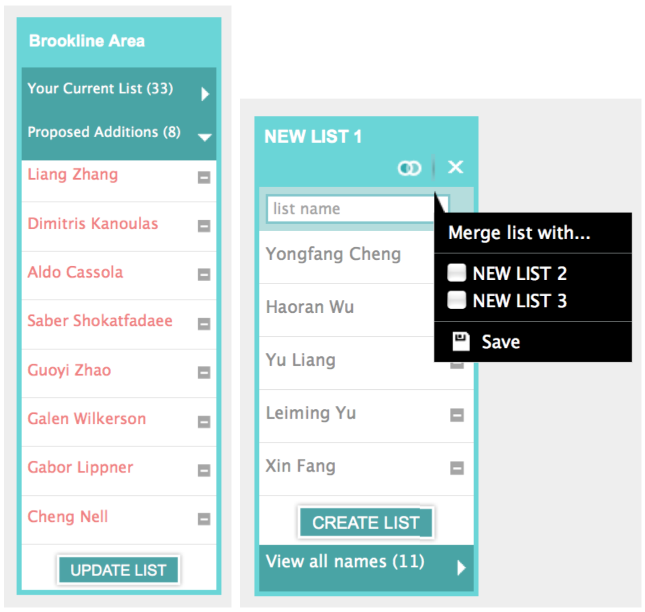
We detect communities for each of them using three different algorithms: BGLL, CNM [2], and Walktrap [11]. We present the cumulative distribution of the maximal normalized conductance per user for Facebook New Orleans in Figure 1 (results for the other networks are similar). Figure 1 shows that almost 40 % of the communities that the algorithms detect have normalized conductance value more than 0.2, indicating that the algorithms are able to find reasonably good quality communities for many users. Also, we notice that all three of these algorithms find communities of similar quality. So, we do not compromise on the quality of the communities by choosing the BGLL algorithm. When looking at the community quality of detected communities at a per-user level, we observe that a majority of users (56%) have at least 10% of their total number of communities with normalized conductance greater than 0.2. This suggests that for most users, there are a few communities that are tightly knit in their local neighborhood.

Figure 2 presents a cumulative distribution of the community sizes that are found; this further indicates that BGLL is able to find small communities. At a per-user level, we find that 94% of users have a median community size less than 20.

Since we observe that automatically detected communities have good conductance values, they are likely to represent small groups of closely knit friends. This observation is compatible with our intuition that these communities can be directly converted into meaningful and useful friendlists.

3. DEMO: FRIENDLIST MANAGER

We now present the user interface of Friendlist Manager and explain the functionalities we provide for Facebook users.



Brookline Area

Your Current List (33)

Proposed Additions (8)

Liang Zhang
Dimitris Kanoulas
Aldo Cassola
Saber Shokatfadaee
Guoyi Zhao
Galen Wilkerson
Gabor Lippner
Cheng Nell

UPDATE LIST

NEW LIST 1

list name

Yongfang Cheng
Haoran Wu
Yu Liang
Leiming Yu
Xin Fang

CREATE LIST

View all names (11)

(a) Existing friendlist (b) New friendlist

Figure 4: Screenshot of (a) an existing friendlist with proposed additions and (b) a newly proposed friendlist with merge friendlist functionality.

We then describe the results from a small deployment of the application to the authors of this paper.

3.1 User interface

The Facebook application Friendlist Manager is available at `http://apps.facebook.com/friendlist_manager`. The application first requests permissions to access the user's data, and then downloads and analyzes the user's local social network. The user is then taken through two steps. In the first step, if the user has existing friendlists on Facebook, Friendlist Manager proposes new additions to each existing friendlist. The user can update existing friendlists with any of the proposed additions, or can skip to the next step directly. In the second step, the user is presented with a set of proposed new friendlists. A screenshot of our application in step two is shown in Figure 3. The user is allowed to remove users from friendlists or drag-and-drop users between friendlists in both steps.

Figure 4(a) shows the interface for proposed additions to existing friendlists, with the following functionalities:

- **Collapsible tab groups** We present two tabs within each friendlist box – one showing the existing members of the friendlist and the second showing the proposed new additions to that friendlist. The newly proposed members are shown in red.

- **Delete members** The user can delete any members shown in the friendlist box (both newly proposed additions and existing members) by clicking the minus button next to each member.

- **Drag-and-drop friend** allows the user to drag one friend out of the friendlist and drop into another friendlist, if the user feels that this friend should belong to another friendlist.

User	Friends	Links between friends	Friendlists detected by Friendlist Manager	Friendlists created by user	Total time in Friendlist Manager
1	51	126	6	5	2.4 min
2	121	608	12	9	5.1 min
3	613	18,820	20	9	4.3 min
4	94	622	6	5	8.8 min
5	149	1,219	5	3	1.1 min

Table 2: Topological characteristics of authors' 1-hop Facebook subgraph along with number of friendlists detected by Friendlist Manager, number of friendlists finally users create on Facebook and time taken by users to use Friendlist Manager for modifying and creating these friendlists on Facebook.

- **Update friendlist** Once the user has finished editing the friendlist, she can save the friendlist to Facebook by clicking "Update List" button at the bottom of each friendlist.

Figure 4(b) shows the interface for newly proposed friendlists, with the following functionalities:

- **Merge friendlists** The user can merge two or more of the proposed friendlists by selecting the merge button on one friendlist and selecting others from a drop-down menu.
- **Delete friendlists** The user can delete any friendlist which is not meaningful.
- **Delete members** This is the same feature as provided in step one.
- **Drag-and-drop friend** This is the same feature as provided in step one.
- **Create friendlist** Create the friendlist on Facebook.

Here, we give more explanation about the Friendlist Manager's functionality: First, in each proposed friendlist, the order of the friends is based on the number of common friends with the user, meaning that the first friend in each friendlist has the highest number of common friends with the user. This results in an ordering of members in each friendlist from most-strongly-connected to least-strongly-connected. Second, the per-friend delete button and the drag-and-drop functionality allow the user to edit the proposed friendlists in a precise way, correcting all errors in assignment that are made by the algorithm. Third, in each friendlist, the two buttons "View all names" and "Hide some names" are used to make the interface more clear for users (especially when users have many friends resulting in large proposed friendlists). Finally, the button "Create List" automatically creates the proposed friendlist on Facebook; after it is created, the friendlist disappears from the screen.

More information about using Friendlist Manager is available at: `http://friendlist-manager.mpi-sws.org/`.

3.2 Experimental evaluation

Next, we briefly describe the results of the Friendlist Manager application deployed to the five authors of this paper. In this evaluation, we focus on the new friendlists created by each user. We present some basic information on the authors' social networks in Table 2. From Table 2, we observe that totally all 5 authors created 31 friendlists on their Facebook accounts and on average each user created 69% of the friendlists proposed by Friendlist Manager. Moreover, each user spent an average of 4 minutes to create all their friendlists.

3.3 Future work

We are exploring mechanisms for automatically suggesting names for the friendlists, further removing the burden on the user. Also, we are actively improving the user interface to make it more intuitive and easy to understand.

4. REFERENCES

[1] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, (10), 2008.

[2] A. Clauset, M. E. J. Newman, , and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70:1– 6, 2004.

[3] Facebook Statistics. `http://www.facebook.com/press/info.php?statistics`.

[4] Facebook lists. `https://www.facebook.com/help/friends/lists`.

[5] Facebook smart lists. `http://techcrunch.com/2011/09/13/facebook-officially-unveils-smart-friend-lists/`.

[6] S. Fortunato. Community detection in graphs. *Physics Reports*, 486:75, 2010.

[7] Y. Liu, K. Gummadi, B. Krishnamurthy, and A. Mislove. Analyzing Facebook privacy settings: User expectations vs. reality. In *Proc. ACM/USENIX Internet Measurement Conference (IMC)*, 2011.

[8] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel. You are who you know: Inferring user profiles in online social networks. In *ACM International Conference on Web Search and Data Mining (WSDM)*, 2010.

[9] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6), 2004.

[10] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[11] P. Pons and M. Latapy. Computing communities in large networks using random walks. *Journal of Graph Algorithms and Applications*, 10:284–293, 2004.

[12] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences*, 101(9):2658, 2004.

[13] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi. On the Evolution of User Interaction in Facebook. In *Proc. ACM SIGCOMM Workshop on Social Networks (WOSN)*, 2009.

[14] C. Wilson, B. Boe, A. Sala, K. P. N. Puttaswamy, and B. Y. Zhao. User Interactions in Social Networks and their Implications. In *Proc. European Conference on Computer Systems (EuroSys)*, 2009.