ReGroup: Interactive Machine Learning for On-Demand Group Creation in Social Networks

Saleema Amershi, James Fogarty, Daniel S. Weld
Computer Science & Engineering
DUB Group
University of Washington
{samershi, jfogarty, weld} @ cs.washington.edu

ABSTRACT
We present ReGroup, a novel end-user interactive machine learning system for helping people create custom, on-demand groups in online social networks. As a person adds members to a group, ReGroup iteratively learns a probabilistic model of group membership specific to that group. ReGroup then uses its currently learned model to suggest additional members and group characteristics for filtering. Our evaluation shows that ReGroup is effective for helping people create large and varied groups, whereas traditional methods (searching by name or selecting from an alphabetical list) are better suited for small groups whose members can be easily recalled by name. By facilitating on-demand group creation, ReGroup can enable in-context sharing and potentially encourage better online privacy practices. In addition, applying interactive machine learning to social network group creation introduces several challenges for designing effective end-user interaction with machine learning. We identify these challenges and discuss how we address them in ReGroup.

Author Keywords
Interactive machine learning, social network group creation, access control lists, example and feature-based interaction.

ACM Classification Keywords
H.5.2 [Information interfaces and presentation (e.g., HCI)]: User Interfaces.

General Terms
Algorithms, Design, Human Factors.

INTRODUCTION
Social networking sites present a conflict between our desire to share our personal lives online and our concerns about personal privacy [13, 33,37]. Well-known examples of privacy blunders include oversharing embarrassing photos with potential employers [19], accidently posting intimate conversations [30], and being fired for publicly criticizing a boss [4]. Major sites have therefore begun advocating customizable friend groups as the latest tool for helping us control with whom we share [27,31].

The currently-advocated approach to custom group creation is to pre-categorize friends in advance of sharing decisions. For example, Google+ requires friends be manually organized into "Circles" before content can be shared with them [31]. Katango [17] and Facebook's "Smart Lists" [27] attempt to aid group curation by automatically generating potential groups based on a person's social graph or common features (e.g., inferred closeness, current city). These automatically generated groups can then be manually edited for correctness or to capture other preferences.

Pre-defined groups may suffice for filtering update streams and news feeds, but usable security recommendations argue that privacy controls for sharing content should operate in context of that content [20,29,40,43]. This is because prior research has shown that willingness to share varies widely based on both content recipients and the content itself [26]. For example, a person's definition of a "close friend" may change when sharing a personal photograph versus inviting people to a party. Furthermore, Jones and O'Neill [16] recently showed that groups created for generic purposes only partially overlap with in-context sharing decisions. Ill-conceived groupings can therefore lead to information leakage, over-restriction, or additional work to revise pre-defined groups in-context.

We present ReGroup (Rapid and Explicit Grouping), a novel system that uses end-user interactive machine learning to help people create custom, on-demand groups in Facebook. As ReGroup (Figure 1) observes a person's normal interaction of adding members to a group, it learns a probabilistic model of group membership in order to suggest both additional members and group characteristics for filtering a friend list. ReGroup differs from prior group member suggestion systems (e.g., Gmail's "Don't forget Bob!" [11] and FeedMe [3]) in its ability to continually update its membership model based on interactive user feedback. As a result, ReGroup can tailor its suggestions to the group a person is currently trying to create (instead of being limited to making suggestions based on a predefined and static notion of similarity or interest). Facilitating on-demand creation of contextualized groups may help encourage better privacy practices in social networks.

CHI'12, May 5–10, 2012, Austin, Texas, USA.

Figure 1. ReGroup uses end-user interactive machine learning to help people create custom, on-demand groups. As a person selects group members (in the *Selected* display), ReGroup suggests additional members (in the *Suggestions* display) and suggests group characteristics as filters for narrowing down a friend list (see five suggested filters at the top of the *Filters* display).

This paper makes the following contributions:

- A new approach to social access control – using end-user interactive machine learning to help people create custom groups on-demand in the context of sharing decisions.
- A discussion of several new challenges for the design of effective end-user interaction with machine learning systems, as exposed in our application to social networks.
- Novel example and feature-based interaction techniques for addressing the above design challenges in ReGroup.
- An evaluation of ReGroup compared to traditional methods of on-demand group creation. Our quantitative and qualitative analyses indicate that different techniques are effective for different types of groups and therefore integrating all techniques in online social networks can support a wider range of desired groups.

REGROUP

ReGroup uses end-user interactive machine learning to help people create custom groups on-demand. In this section, we first use an example to illustrate how a person can create a group with ReGroup. We then discuss the challenges inherent to interactive machine learning for group creation and how we address them in our design of ReGroup.

Example Usage Scenario

Ada wants to advertise a confidential research talk to relevant friends at the University of Washington, so she decides to use ReGroup to create a private group for the ad. To start, she thinks of a friend she knows will be interested in the talk, searches for them by name (via a search box, left in Figure 1) and adds them to her group (*Selected* display, top in Figure 1). ReGroup learns from this example and then tries to help Ada find other friends to include. It

re-organizes her friend list to sort relevant friends to the top of the *Suggestions* display (right in Figure 1). Ada now sees several additional friends she wants to include. She adds them to her group all at once by drag-selecting and then clicking the *Add Selected* button (right in Figure 1). With these additional examples, ReGroup learns more about the group being created and again re-organizes Ada's friends to help her find more group members.

As ReGroup learns about the group Ada is creating, it also presents relevant group characteristics she can use as filters. For example, given the currently selected friends, ReGroup believes Ada might want to include other people that live in Washington, that have several mutual friends with her, or that work at the University of Washington (top of *Filters* display in Figure 1). Although not everybody Ada wants to include works at the University of Washington (e.g., some are students), she agrees that they all likely live in Washington. She clicks the "Washington" filter, causing ReGroup to remove friends who do not live in Washington. This helps Ada by reducing the number of people she must consider when finding friends to include in her group.

Ada continues interacting with ReGroup this way, explicitly adding group members and filtering when necessary, until she has included everyone to whom she wants to advertise.

Identifying Features

ReGroup's ability to suggest group members is powered by its interactive machine learning component. Machine learning works by discovering patterns in examples. A system is therefore strongly influenced by the quality of information contained in the representations of those examples (i.e., the features). We based our features on related work on social networks (e.g., [11]). We also conducted an online survey of Facebook and Google+ users to identify common groups we could support. We distributed the survey to our own Facebook and Google+ contacts, obtaining 69 responses (21 Facebook and 48 Google+) describing 244 customized groups (32 Facebook and 212 Google+). Facebook and Google+ advocate creating groups in advance, so the groups identified in our survey may not correspond to groups people would create on-demand (i.e., the composition of permanent groups may differ from groups created in context). Nevertheless, these gave a starting point from which to distill potential features.

Table 1 presents the 18 features currently used by ReGroup, each with an example group the feature might help support. In this research, we endeavored to support common groups as part of demonstrating the potential for interactive machine learning in social access control. Our features are therefore not intended to be exhaustive.

Suggesting People while Preserving End-User Control

ReGroup's group member suggestions are realized by a Naïve Bayes classifier [5]. Each friend is represented by a set of n feature-value pairs (Table 1). The probability of each friend being a member of the group g is then computed via the following Bayesian formulation:

Feature	Description/Examples
Gender, Age Range	"Sorority friends", "People my age"
Family Member	"People I'm related to"
Home City, State and Country	"Friends from my hometown"
Current City, State, Country	"Anyone I know who lives nearby"
High School, College, Graduate School	"Anyone from my department at school"
Workplace	"CoWorkers"
Amount of Correspondence	Number of Inbox, Outbox and Wall messages sent and received. "People I socialize with regularly"
Recency of Correspondence	Time since latest correspondence. "Friends I currently interact with"
Friendship Duration	Time since first correspondence [11]. "Friends I've known over 10 years"
Number of Mutual Friends	"My group of closest friends"
Amount Seen Together	Number of photos/events a person and friend are both tagged in. "People I see on a regular bases"

Table 1. The 18 features currently used by ReGroup's interactive machine learning component.

$$P(g|x_1, x_2, \ldots, x_n) = \frac{P(x_1, x_2, \ldots, x_n|g)P(g)}{\sum_{g'=g,\bar{g}} P(x_1, x_2, \ldots, x_n|g')P(g')}$$

where $P(x_1, x_2, \ldots, x_n|g)$ is the likelihood of a friend with feature values $x_1, x_2, \ldots, x_n$ being a member of g, $P(g)$ is the prior probability of any friend belonging to g, and the denominator is the probability of seeing the set of feature values in the data and serves as a normalizing constant.

The Naïve Bayes assumption considers each feature to be conditionally independent given the class, reducing the likelihood computation to:

$$P(x_1, x_2, \ldots, x_n|g) = \prod_{i=1}^{n} P(x_i|g)$$

where the probability of a group member having a particular feature value, $P(x_i|g)$, can be estimated by a frequency count of the number of current group members having that feature value over the total number of current group members. We also use Laplace smoothing to improve performance in the presence of limited training data. Although the independence assumption is often violated in real-world data, Naïve Bayes has been shown to work well in many practical applications [5]. Naïve Bayes also gracefully handles missing data and allows for a straightforward interpretation of features, both important aspects of ReGroup as will be discussed below.

ReGroup's classifier is re-trained every time a person adds friends to a group. ReGroup then reorders a person's remaining friends according to who is most likely to also belong to the group as computed by the updated classifier.

To preserve end-user control during interaction with the classifier, we made the design decision that people must *explicitly* approve every group member, as opposed to having ReGroup automatically add suggested friends.

Indirectly Training an Effective Classifier

Effective training of machine learning systems requires both positive and negative examples. Therefore, a system focused on *training an effective classifier* will be designed to solicit explicit positive and negative examples (e.g., [1]). In our case, however, a person's primary goal is to *create a group*. That is, the classifier is a disposable side effect and a person is not concerned about its generalization.

To mitigate the effects of people primarily providing positive examples, we designed ReGroup to obtain implicit negative examples during interaction. When a person selects a group member from an ordered list of friends, ReGroup increases the probability that the skipped friends (i.e., friends preceding the selection in the ordered list) are *not* intended for the group. This is achieved by assigning the preceding friends implicit negative labels in ReGroup's group membership computations. However, as this heuristic is not always correct (e.g., a person's gaze may be drawn to a friend further down in the list without having viewed or decided upon the preceding friends), an implicit negative example contributes only a partial frequency count, α, in ReGroup's computation of $P(x|g)$. ReGroup also still includes implicitly labeled friends in its list of suggestions. We found that setting α=0.2*n, where n is the number of times a friend is skipped, worked well in practice.

Unlearnable Groups

While experimenting with early versions of ReGroup, we observed that skipped friends would sometimes continue bubbling back up in the ordered list during group creation. Further examination revealed that this occurred when (1) a friend had all the characteristics of the group, as modeled by the classifier, but (2) was skipped for some reason not captured by the system (e.g., a person planning a surprise party would obviously not invite the guest of honor). Implicit negative examples are not powerful enough to prevent repeated suggestion of such friends because of their high similarity to the many positive examples. This problem occurs in all machine learning systems when the hypothesis language is insufficiently expressive to model the true concept. However, it can be particularly frustrating in an interactive system like ReGroup, as a person can quickly grow tired of repeatedly skipping the same friend. We addressed this with an explicit penalty term, v, in our group membership estimation as follows:

$$P(g|x) = P(g|x) * v^n$$

where n is the number of times a friend was skipped. Our preliminary experiments showed that setting v=0.9 achieved the desired effect (i.e., it reduced the likelihood of a skipped friend continuing to be highly ranked when ReGroup could not learn the true concept).

Integrating Knowledge about Group Members via Decision-Theoretic Filter Suggestions

Most end-user interactive machine learning systems focus only on interaction with examples (e.g., [8, 10]). However, people often have knowledge about the shared properties of groups. We hypothesized that enabling interaction with those features might accelerate the group creation process. We chose to realize end-user interaction with features in ReGroup using faceted search [15], a popular method for helping people find items in collections (e.g., [46]). With faceted search, people find items by filtering a collection based on feature values. ReGroup provides two ways for people to filter friends by feature values:

- Via a suggested list of five top feature value filters (top of *Filters* display, left in Figure 1).
- Via a static, hierarchical list of all feature value filters (bottom of *Filters* display, left in Figure 1).

Previous methods for ordering filters have used metrics like hit count [15] or query keyword similarity [46]. However, none of these are appropriate for our domain. We therefore formulate the problem decision-theoretically. Intuitively, we want ReGroup to suggest filters that will reduce effort required during group creation. A suggested filter must therefore represent the intended group well (i.e., $P(x_i|\, g)$ must be high). However, this alone does not guarantee that a filter will prune unwanted friends. We therefore combine this with the expected utility of a filter as follows:

$$U(x_i) = P(x_i|\, g) * InformationGain(X_i)$$

Here, we use information gain to approximate the potential time savings of activating a filter [25]. We use this formulation in choosing the top five filters to suggest, as well as in ordering feature values within the static list.

Missing Data

Missing data, which can lead to unpredictable behavior in machine learning systems, is rampant in the domain of online social networks. People choose not to supply information, have strict privacy settings, or are inconsistent with their online activity. The Naïve Bayes classifier gracefully handles missing data by ignoring features with missing values when computing group membership probabilities. However, missing data can still adversely impact the usefulness of filters by leading to incorrectly filtered friends (e.g., when filtering on "Seattle", a friend who lives in Seattle might be filtered because they have not supplied this to Facebook) or incorrectly preserved friends (e.g., when filtering on "Seattle", retaining a friend who lives in Austin but has not supplied this to Facebook).

ReGroup attempts to estimate missing values by creating additional feature classifiers for predicting missing values conditioned on all other available features. Then, when applying a filter, ReGroup only eliminates friends who are *guaranteed* to be ineligible (i.e., have provided some value other than the filter value). For friends with missing data, ReGroup indicates its uncertainty by displaying a question

mark under their name (see Figure 1). If a person hovers over a question mark, ReGroup displays a tooltip showing its guess for that friend's corresponding feature value. ReGroup thus avoids incorrectly eliminating friends while reducing irritation by explicitly displaying its guesses.

Implementation Details

ReGroup is implemented using Facebook's Application Platform [7] and a Firefox Greasemonkey script [12]. ReGroup uses Facebook's Application Platform to access a person's relevant information (see Table 1), while the Greasmonkey script allows ReGroup to run within a person's own Facebook account. ReGroup is not publicly available (i.e., only our study participants could access the Facebook Application and use the Greasemonkey script installed on the study computer). ReGroup accessed a participant's Facebook information only after they provided explicit and informed consent and only for enabling its interface, staying within Facebook's Terms of Service.

EVALUATION

We conducted an evaluation to explore the tradeoffs between end-user interactive machine learning for on-demand custom group creation and Facebook's current approach of allowing manual selection from an alphabetical list or searching by name [39]. We also wanted to compare a design including our feature-based interaction with a more typical design using only example-based interaction.

Interface Conditions

We evaluated the following interfaces:

- *Alphabet*. A person can search by name or scroll through an alphabetical list to find friends. This is equivalent to Facebook's current on-demand group creation process.
- *Example-Only*. Each time a person adds a friend to the group, the list of friends is reordered based on ReGroup's current estimation of group membership probability. People can also still search for friends by name.
- *Example-Attribute*. The full ReGroup design, enhancing the *Example-Only* with our decision-theoretic technique for feature-based interaction.

Design and Procedure

We ran a within-subjects study, counterbalancing order of interface conditions using a Latin square design. At the beginning of the study, we told participants they would be testing new tools for helping people create custom groups in Facebook. We also explained the tools would work in their own Facebook accounts and that they would be testing the tools by creating groups of their own friends. Participants were then informed about what data would be accessed and how it would be used. We continued only after they provided written consent and granted our Facebook Application permissions to access their data.

Next, prior to seeing any interface, participants were asked to think of six groups they could create during the study. We chose this approach to ensure groups were meaningful, as assigned groups may not correspond to distinctions a person would make among their friends. When thinking of groups, participants were instructed to imagine they were about to post a new comment or share a new photo and only wanted to share it with a select group of friends. We also provided them with a list of ten example groups based on frequent responses to our online survey (e.g., "home town friends", "close friends") to help them decide. Resulting groups ranged from typical (e.g., "Family", "CoWorkers") to more unique and nuanced (e.g., "Older Faculty Members", "People I Care About"). We also asked participants to estimate group size ("small", "medium" or "large"). The experimenter then sorted groups by estimated size and assigned them to conditions in order (thus roughly balancing groups across conditions by size).

The experimenter then demonstrated ReGroup (using the full *Example-Attribute* interface) with the experimenter's own Facebook account. Participants were told they would use three variations of the ReGroup interface. Before each condition, the participant practiced with the corresponding interface by creating a simple gender-based group. After the practice, participants used the interface condition to create two of their groups (they were not told which groups they would create until they were about to create them). To avoid exhausting participants, we limited each group to a maximum of 4 minutes (we did not inform participants of this limit, but simply asked them to stop if they reached it).

All interface actions were time-stamped and logged. All participant information and logged data was stored anonymously, using unique identifiers, on a secure server accessible only by the researchers. After each group, participants completed a short questionnaire containing 5-point Likert scales (1=strongly disagree, 5=strongly agree) about the group they just created (e.g., "I was happy with the group I just created"). At the end of the study, participants filled out a final questionnaire to comment on their overall experience and compare the interfaces (e.g., "Rank the versions to indicate which was your favorite"). The study lasted 1 hour and participants were given a $20 Amazon gift certificate for their participation.

Participants and Data

Twelve people (four female, ranging in age from 18-34) were recruited via a call for participation sent to several university mailing lists. As a result, all of our participants were technically savvy, but ranged from undergraduates to staff members from a variety of disciplines (e.g., Computer Science, Psychology, Design). We required participants to have an active Facebook account in use for at least one year and to have at least 100 friends. This was to help ensure enough activity was available to enable ReGroup's suggestions. As expected, participants varied in their composition and number of friends (mean=385.4, min=136, max=781) and their Facebook activity (ranging from every day to a few times per month). On average, our participants also had 36.3% missing data with respect to ReGroup's features.

RESULTS

We performed all of our log and Likert scale data analyses using a nonparametric repeated measures analysis of variance, after aligning the data according to the *aligned rank transform* [44] to preserve interaction effects due to having participants create two groups per condition. We also performed post-hoc pairwise comparisons when a significant effect was observed. To analyze our final ranking questions, we used a randomization test of goodness-of-fit [23] which is more robust against smaller sample sizes than a standard Chi-Square test. For each test, we ran 10,000 Monte Carlo simulations. Tables 2 and 3 show the per-condition means and standard deviations for all metrics used in our log and Likert scale data analyses, respectively. Table 3 also shows the number of participants choosing each condition for each of our ranking questions. We discuss all of our quantitative analyses in the context of our qualitative observations and feedback from participants.

We analyze our study data in terms of the overall time taken and final group sizes, the speed and effort of selecting group members, and interface element usage. Note that we cannot evaluate group accuracy because no adequate ground truth is available or obtainable. Asking participants to re-create Facebook or Google+ groups could bias their notion of those groups. Alternatively, asking participants to verify group completeness would require exhaustive labeling (recognized as error-prone [38] and analogous to list scrolling in the *Alphabet* condition).

Final Times and Group Sizes

Overall, participants created 72 groups with a total of 2077 friends. Examining the *Final Time* taken to create groups, our analysis shows a significant effect of interface condition ($F_{2,55}$=6.95, p≈.002). Post-hoc pairwise analyses reveal that participants using the *Alphabet* interface took significantly less time to create groups than when using both the *Example-Only* ($F_{1,55}$=5.93, p≈.018) and *Example-Attribute* ($F_{1,55}$=13.4, p≈.0006) interfaces. There was no difference in *Final Time* between *Example-Only* and *Example-Attribute* conditions. One explanation for participants taking less time in the *Alphabet* condition is that both the reordering conditions (*Example-Only* and *Example-Attribute*) required additional time to update the display when reordering or filtering. Another contributing factor could be that participants in the *Alphabet* condition often attempted to recall friends by name to avoid scrolling through their entire list of friends (e.g., *"I was surprised how useless alphabetical ordering is, but keyword search was very useful"*). This may have resulted in people forgetting to include some friends and stopping early. Interestingly, participants who resorted to scrolling through the full list often felt like they missed people (e.g., *"umm, I guess that's it"*, *"there's probably more, but oh well"*). One participant also explicitly commented *"It's too easy to forget about people when it's ordered alphabetically."*

	Alphabet	***Example-Only***	***Example-Attribute***
*Final Time**	163.0/63.4s	196.9/56.8s	216.0/35.8s
Final Group Size	25.3/24.8	34.0/40.6	27.2/22.1
Mean Select Time	8.6/3.9s	13.9/16.3s	15.9/16.5s
*SD Select Time**	9.4/6.5s	13.5/8.5s	19.7/15.1s
*Mean Position**	171.8/87.2	58.1/78.9	38.5/48.8
*SD Position**	101.7/56.1	41.1/33.3	40.5/41.1
Single Selections	18.0/27.0	8.5/6.2	7.9/5.1
*Multi-Selections**	0.4/0.8	2.8/2.9	2.8/3.0
Search-Selections	4.3/4.7	2.1/2.7	2.0/2.1

Table 2. Mean/SDs of all metrics used in our log data analyses. *'s indicate a significant effect was observed.

	Alphabet	***Example-Only***	***Example-Attribute***
Happiness	3.9/0.9	3.9/1.1	4.0/0.7
*Easiness**	2.6/1.0	3.2/1.2	3.3/0.8
*Quickness**	2.3/1.0	3.0/1.2	3.4/1.0
*Favorite**	1	0	11
*Best Helped**	3	1	8

Table 3. Likert mean/SDs (1=strongly disagree, 5=strongly agree) and ranking (number of participants) responses. Metrics with *'s indicate a significant effect was observed.

Difficulty recalling friends in the *Alphabet* interface could have resulted in the shorter *Final Times* in that condition. However, one would then also expect to see larger *Final Group Sizes* in the reordering conditions because of their ability to surface relevant people, favoring recognition over recall (e.g., *"Reordering makes it much faster than alphabetical because it reminds you of people without having to do an exhaustive search"* and *"Reordering helps me quickly pick friends in the first rows. Filters keep me from frequent scrolling"*). However, our analysis showed no significant difference in *Final Group Size* across conditions. Further analysis showed the presence of a ceiling effect in all conditions, suggesting that participants were often cut short of the time they needed to complete their groups, which could account for the lack of difference in *Final Group Size*. This effect was also more pronounced in the reordering conditions (16.7%, 33.3% and 45.8% of interactions were cut short in the *Alphabet*, *Example-Only*, and *Example-Attribute* conditions, respectively).

Speed and Effort in Selecting Group Members

To compare the time between group member selections, we had to account for the fact that participants could add friends individually or in bulk (i.e., a multi-select action). In the case of a multi-select, we assigned the time between the action and the previous group member selection to the first friend in the multi-selection and assigned a time of zero to the rest of the friends. We did not see an effect of interface condition on *Mean Select Time*. We did however see a difference in *SD Select Time* ($F_{2,55}$=7.78, p≈.001), with post-hoc pairwise comparisons showing significant or marginal differences in all cases: *Alphabet* was less than *Example-Only* ($F_{1,55}$=2.83, p≈.09) and *Example-Attribute*

($F_{1,55}$=15.5, p≈.0002), and *Example-Only* was less than *Example-Attribute* ($F_{1,55}$=5.07, p≈.03).

Although we did not see an effect of condition on *Mean Select Time*, the average *Mean Select Time* was smallest in *Alphabet* (see Table 2). This is partially an artifact of the time needed to update the display in the reordering conditions. In *Example-Attribute*, a display update can also occur as a result of a filtering action. As this confounds our analysis of *Select Time*, we decided to measure the position in the display of each group member immediately before selection as a proxy for effort level. That is, if position is low, a friend was closer to the top of the display and therefore required less effort to locate. We saw a significant effect of condition on *Mean Position* ($F_{2,55}$=39.1, p≈.0001), with post-hoc pairwise comparisons showing that the *Mean Position* in *Alphabet* was significantly greater than in both *Example-Only* ($F_{1,55}$=47.8, p≈.0001) and *Example-Attribute* ($F_{1,55}$=67.7, p≈.0001). We also saw a significant difference in consistency of position as measured by *SD Position* ($F_{2,55}$=19.6, p≈.0001), again with *SD Position* being significantly greater in *Alphabet* compared to both *Example-Only* ($F_{1,55}$=2.81, p≈.0001) and *Example-Attribute* ($F_{1,55}$=31.8, p≈.0001). Lower positions indicate that people had to scroll less to search for friends in the reordering conditions because these conditions sorted potential group members closer to the top of the display for easy recognition and access. As expected, the *Mean Position* of selected friends in the *Alphabet* condition was roughly half, 44.6%, of the average number of Facebook friends of our participants and their *SD Position* was highly varied because group members were often evenly distributed throughout the entire alphabetical list.

Our questionnaire results provide additional evidence of reduced effort in the reordering conditions. Analyses of our Likert scale questions show an effect of condition on perceived levels of *Easiness* ($F_{2,55}$=4.33, p≈.018), with the both the *Example-Only* and *Example-Attribute* interfaces being perceived as easier to use than the *Alphabet* interface ($F_{1,55}$=5.74, p≈.02 and $F_{1,55}$=7.18, p≈.01, respectively). Similarly, we found a significant effect of condition on perceived *Quickness* ($F_{2,55}$=6.63, p≈.003) in creating groups, again with *Example-Only* and *Example-Attribute* interfaces being perceived as quicker than *Alphabet* ($F_{1,55}$=5.80, p≈.02 and $F_{1,55}$=12.74, p≈.0008, respectively). We saw no difference in terms of perceived *Easiness* or *Quickness* between *Example-Only* and *Example-Attribute.* Perceived *Easiness* and *Quickness* in the reordering conditions is likely due to these interfaces automatically surfacing relevant friends (e.g., "*Sometimes it really looked as if the system was reading my mind!"*).

Behavioral Differences and Feature Usage

In all interfaces, participants could select friends to include one at a time (*Single Selections*), several at a time (*Multi-Selections*), or by searching for them by name (*Search-Selections*). Our participants used all of these features in each condition, however they searched for friends by name and added friends one by one less often in the reordering conditions compared to the *Alphabet* condition. In addition, we found a significant effect of condition on the number of *Multi-Selections* ($F_{2,55}$=10.9, p≈.0001) with the *Example-Only* and *Example-Attribute* conditions both showing increased *Multi-Selections* compared to the *Alphabet* condition ($F_{1,55}$=17.4, p≈.0001 and $F_{1,55}$=15.4, p≈.0002, respectively). Increased *Multi-Selections* and fewer *Single* and *Search-Selections* in the reordering conditions is likely due to these interfaces making friends easier to add as a group by sorting relevant friends to the top of the display and filtering out irrelevant friends (in *Example-Attribute*).

Our logged data also showed that participants used the suggested and static filters when they were available (in *Example-Attribute*). Participants selected *Suggested Filters* 1.9 times on average (*SD*=1.3), selected *Static Filters* 0.9 times on average (*SD*=1.3), and *Unselected Filters* that they had previously selected 2.3 times on average (*SD*=2.8). Our observations showed that participants sometimes used the filters in clever and unexpected ways, such as selecting them temporarily to find a certain set of friends and then unselecting them to find others. One participant commented that the "*Filters helped me guide the tool"*. Other comments indicate that the suggested filters served as an explanation of how the system was working (e.g., *"The filters helped me understand what patterns the tool was discovering, and thus helped understand the tool's behavior"*).

Discussion

Overall, our evaluation showed that different interfaces worked well for different kinds of groups (e.g., *"[My favorite version] depended on the group I was creating"*). We observed, and participants reported, that the *Alphabet* condition appeared to work well for small groups whose members could be easily recalled by name (e.g., family members with the same name, co-authors of a paper, and members of a cross-country team). In contrast, the reordering conditions worked better for larger and more varied groups (e.g., childhood friends, local friends or friends from particular regions of the world, college friends, former and current colleagues, people in a particular field of work, and friends with a professional relationship). One participant noted that for *"most of the small groups that I wanted to create, I already knew the list in my head [so] alphabetical was easiest. However, for larger groups, reordering was most efficient."* A likely result of this is that we found no significant difference in terms of overall participant *Happiness* with their final groups created in each condition.

Although a significant majority of our participants rated the full ReGroup interface (i.e., *Example-Attribute*) as their overall *Favorite* (χ^2=18.5, p≈.0001) and felt that it *Best Helped* them create groups (χ^2=6.5, p≈.04), participants did become frustrated when ReGroup was unable to model the

group well, as this essentially resulted in having to search through an unordered list for group members (e.g., *"Ordering is useful for some lists, annoying for others"* and "*though I liked the filters, I think they didn't work in some cases"*). As with all machine learning based systems, the ability to model a concept is highly dependent upon the quality of the underlying data (e.g., the expressiveness of the features and the availability of data). Interestingly, some participants could tell why ReGroup was not working well in some cases (e.g., *"Groups that all have something in common that is tracked by Facebook (i.e., college) were easier to create than a group of people who are seemingly unrelated based on Facebook information."*).

RELATED WORK

Privacy continues to be a concern in online social networks [13,33,37]. Traditional mechanisms for specifying social access control lists have been too inexpressive (e.g., pre-defining lists of "Friends" or "Everyone" in Facebook) or tedious and error-prone (e.g., manual selection from an alphabetical list of friends [28,38]). This has motivated research on alternative methods. For example, Toomim *et al.* examine guard questions testing for shared knowledge [41] and McCallum *et al.* examine automatically inferring access control roles from email communication [24].

Recent interest has developed in automatic detection of communities within social networks (e.g., [16,17,22,27]). The assumption in such work is that detected communities will overlap with a person's eventual sharing needs. SocialFlows [22] automatically creates hierarchical and potentially overlapping groups based on email communication. These groups can then be edited via direct manipulation and exported to Facebook and Gmail. Katango [27] and Facebook's "Smart Lists" [17] feature also follow this model by providing a set of automatically generated friend groups which can then be manually edited for correctness. Jones and O'Neill [16] recently evaluated the feasibility of this approach by comparing manually defined groups with automatically detected network clusters. They found their automatically-detected clusters only overlapped with manually-defined groups by 66.9% on average, suggesting a need for considerable manual refinement. After following up with their participants, they further discovered that predefined groups only partially correspond to in-context sharing decisions (77.8-90.8% overlap with manually defined groups and only 33.8-76.1% overlap with automatically generated groups). In contrast, our focus on custom, on-demand group specification via end-user interactive machine learning is better aligned with usable security recommendations [20,29,40,43].

Prior work facilitating on-demand sharing or access control in social networks includes Gilbert and Karahalios's research on modeling relationship or tie strength between individuals based on network structure, communication, and shared profile data [11]. Predicted tie strength could potentially inform privacy levels for protected content. In contrast to modeling relationships, Bernstein *et al.*'s FeedMe system [3] models user interest using term-vector profiles in order to recommend people with which to share particular pieces of content. Our work differs from these in that ReGroup iteratively learns a model based on end-user provided examples and feedback. By iteratively updating the underlying model, ReGroup can tailor its group member recommendations to the specific group a person is currently trying to create. More similar to our approach of defining groups by example is Gmail's "Don't forget Bob!" feature, which recommends additional email recipients based on a few seed contacts and a social graph constructed from previous email [34]. However, ReGroup provides more control over its suggestions by continuing to iteratively update its underlying model based on further examples.

Our work also contributes to recent research on designing end-user interactive machine learning systems (e.g., [8,10]). Most similar to our work is Fang and LeFevre's interactive machine learning-based privacy wizard for online social networks [9]. Their wizard employs active learning to construct a privacy classifier that maps item-friend pairs to allow or deny actions. ReGroup differs in that we use end-user interactive machine learning as a tool for helping people select members of a group rather than for creating a robust classifier. As a result, ReGroup's design aims to balance end-user flexibility and control with speed, as opposed to an active learning approach that prioritizes speed at the potential expense of user experience [2]. Furthermore, ReGroup preserves end-user desired control of group membership [14] by requiring explicit approval of group members instead of delegating control to a classifier.

Defining groups via member examples and interaction with group characteristics is related to work on example and feature-based feedback in interactive machine learning systems [6,32,45]. Each of these explored the utility of feature or example-and-feature-based feedback for creating document classifiers. Wong *et al.* [45] allowed end-users to explicitly define relevant features by highlighting words used to update a classifier model. Other work [6,32] has taken an active learning approach to soliciting end-user labels of features and examples. Our approach differs in that we do not solicit feature-level feedback for the purpose of training a classifier. Rather, we enable feature interaction for the purpose of helping people search for relevant examples. Our approach uses a novel technique of integrating human knowledge of features into the interactive machine learning process through filtering on group characteristics. This makes our work more similar to work on faceted search and interaction [15].

Secord *et al.* [35] explored example and facet-based interaction for creating collections of music (i.e., playlists). In their system, end-users could both view songs related to given examples or enter textual queries (e.g., "lots of rock") that were then mapped to facet values (e.g., genre=rock) and used as constraints on song recommendations. Their

rationale for free form, text-based facet interaction was that songs have potentially hundreds of facets and traditional static presentations of facets would be too cumbersome for interaction. The social network domain can also have an overabundance of facets, however our approach is to highlight and rank facets highly relevant to the intended group. This approach favors recognition of relevant facets instead of relying upon recall, which can be particularly useful when facets are vague or difficult to define.

Previous approaches to facet ranking for search result exploration have based ranking on hit count [15], query keywords [46], query logs and click through data [42], or user models defined by explicit ratings of documents [18]. We base ranking on statistical regularities of the currently selected group members. Most similar to our approach is Li *et al.*'s Facetedpedia [21], which extracts and ranks facets based on the top results of a keyword search. However, their ranking is based on the navigation cost of reaching the top results and therefore serves more as a sense-making tool rather than a tool to search for additional items.

CONCLUSION AND FUTURE WORK

This paper presented ReGroup, a novel system that employs end-user interactive machine learning to help people create custom groups on-demand in online social networks. Our evaluation showed that different group creation techniques helped in creating different kinds of groups. For example, the traditional search-by-name approach is efficient for small, well-defined groups where members are easy to recall by name. ReGroup's support for recognition over recall, embodied by both its group member and filter suggestions, helps for larger and more varied groups. Therefore, we advocate integrating ReGroup's novel techniques with traditional methods used in online social networks to support a wider range of potential groups.

This paper also discussed the design challenges associated with interactive machine learning in social access control and detailed how we addressed these challenges in ReGroup. One challenge was in addressing unlearnable groups, typically the result of missing information. Although we attempted to address this issue with our technique for group member prediction, it still caused some frustrations during our evaluation. Additional features should improve ReGroup's performance, but missing data is inherent to social networks. Future work might therefore aim to detect such cases and degrade gracefully. Another design decision was to focus end-user attention on creating groups, rather than on training a robust classifier. However, creating a classifier as a by-product of group creation could enable automatic group maintenance as relationships change over time. In the future we hope to evaluate ReGroup's classifiers for group maintenance and to design additional interaction techniques to support this task.

ACKNOWLEDGMENTS

We thank Michael Toomim for his thoughtful insights and Jacob O. Wobbrock and Leah Findlater for their guidance on our analyses. This work was supported in part by the National Science Foundation under awards IIS-0812590, IIS-1016713, and OAI-1028195, by Office of Naval Research award N00014-06-1-0147, by the WRF / TJ Cable Professorship, and by a Google Faculty Research Award.

REFERENCES

1. Amershi, S., Fogarty, J., Kapoor, A. and Tan, D. Overview-Based Example Selection in Mixed-Initiative Interactive Concept Learning. In *Proc. UIST 2009*, ACM Press (2009), 247-256.
2. Baum, E.B. and Lang, K. Query Learning can work Poorly when a Human Oracle is Used. In *Proc. IJCNN 1992,* IEEE Computer Society (1992), 609-614.
3. Bernstein, M., Marcus, A., Karger, D.R. and Miller, R. C. Enhancing Directed Content Sharing on the Web. In *Proc. CHI 2010*, ACM Press (2010), 971-980.
4. Chen, S. Can Facebook get you Fired? Playing it Safe in the Social Media World. Nov. 10, 2010. http://articles.cnn.com/2010-11-10/living/facebook.fired.social.media.etiquette_1_social-media-worker-posts-workplace-complaints.
5. Domingos, P. and Pazzani, M. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning 29*, (1997), 103-130.
6. Druck, G., Settles, B. and McCallum, A. Active learning by labeling features. In *Proc. EMNLP 2009*, 81-90.
7. Facebook Application Platform. http://developers.facebook.com/
8. Fails, J.A. and Olsen, Jr., D.R. Interactive Machine Learning. In *Proc. IUI 2003*, ACM Press (2003), 39-45.
9. Fang, L. and LeFevre, K.. Privacy Wizards for Social Networking Sites. In *Proc. WWW 2010*, ACM Press (2010), 351-360.
10. Fogarty, J., Tan, D., Kapoor, A., and Winder, S. CueFlik: Interactive Concept Learning in Image Search. In *Proc. CHI 2008*, ACM Press (2008), 29-38.
11. Gilbert, E. and Karahalios, K. Predicting Tie Strength with Social Media. In *Proc. CHI 2009*, ACM Press (2009), 211–220.
12. Greasemonkey. http://www.greasespot.net/
13. Gross, R. and Acquisti, A. Information Revelation and Privacy in Online Social Networks. In *Proc. WPES 2005*, ACM Press (2005), 71-80.
14. Halliday, J. Facebook Changes See the Social Network Trying to be More Social. Sept. 21, 2011. http://www.guardian.co.uk/technology/pda/2011/sep/21/facebook-changes-social-network.
15. Hearst, M. Design Recommendations for Hierarchical Faceted Search Interfaces. *SIGIR Workshop on Faceted Search 2006,* ACM Press (2010).

16. Jones, S. and O'Neill, E. Feasibility of Structural Network Clustering for Group-Based Privacy Control in Social Networks. In *Proc. SOUPS 2010,* ACM Press (2010).

17. Kincaid, J. Kleiner-Backed Katango Organizes Your Facebook Friends into Groups for You. July 11, 2011. http://techcrunch.com/2011/07/11/kleiner-backed-katango-automatically-organizes-your-facebook-friends-into-groups-for-you.

18. Koren, J., Zhang, Y. and Liu, X. Personalized Interactive Faceted Search. In *Proc. WWW 2008*, ACM Press (2008), 477-486.

19. Korkki, P. Is Your Online Identity Spoiling Your Chances? Oct. 9, 2010. http://www.nytimes.com/2010/10/10/jobs/10search.html

20. Lederer, S., Hong, J.I., Dey, A.K., and Landay, J.A. Personal Privacy through Understanding and Action: Five Pitfalls for Designers. *Personal Ubiquitous Computing 8*, 6 (2004), 440-454.

21. Li, C., Yan, N., Roy, S.B., Lisham, L. and Das, G. Facetedpedia: Dynamic Generation of Query-Dependent Faceted Interfaces for Wikipedia. In *Proc. WWW 2010,* ACM Press (2010), 651-660.

22. MacLean, D., Hangal, S., Teh, S.K., Lam, M.S. and Heer, J. Groups Without Tears: Mining Social Topologies from Email. In *Proc. IUI 2011*, ACM Press (2011), 83-92.

23. Manly, B.F.J.*Randomization, Bootstrap and Monte Carlo Methods in Biology, 2nd ed*. Chapman & Hall, NY, 1997.

24. McCallum, A., Wang, X. and Corrada-Emmanuel, A. Topic and Role Discovery in Social Networks with Experiments on Enron and Academic Email. *Journal of Artificial Intelligence Research 30*, 1 (2007), 249–272.

25. Mitchell, T. *Machine Learning*. McGraw Hill, 1997.

26. Olson, J.S., Grudin, J. and Horvitz, E. A Study of Preferences for Sharing and Privacy. *Ext. Abstracts CHI 2005*, ACM Press (2005), 1985-1988.

27. Ortutay, B. Facebook to Organize Friends in 'Smart Lists'. Sept. 13, 2011. http://www.usatoday.com/tech/news/story/2011-09-13/facebook-smart-lists/50389154/1.

28. Pachal, P. Google Circles: The Dumbest Thing About Google+. June 29, 2011. http://www.pcmag.com/article2/0,2817,2387808,00.asp.

29. Palen, L. and Dourish, P. Unpacking "Privacy" for a Networked World. In *Proc. CHI 2003*, ACM Press (2003), 129-136.

30. Parker, A. Congressman, Sharp Voice on Twitter, Finds It Can Cut 2 Ways. May, 30. 2011. http://www.nytimes.com/2011/05/31/nyregion/for-rep-anthony-weiner-twitter-has-double-edge.html.

31. Pogue, D. Google+ Improves on Facebook. July 13, 2011. http://www.nytimes.com/2011/07/14/technology/personaltech/google-gets-a-leg-up-on-facebook.html.

32. Raghavan, H. and Allan, J. An Interactive Algorithm for Asking and Incorporating Feature Feedback into Support Vector Machines. In *Proc. SIGIR 2007*, ACM Press (2007), 79-86.

33. Rosenblum, D. What Anyone can Know: The Privacy Risks of Social Networking Sites. *IEEE Security & Privacy 5,* 3 (2007), 40-49.

34. Roth, M. Ben-David, A., Deutscher, D., Flysher, G., Horn, I., Leichtberg, A., Leiser, N., Matias, Y. and Merom, R. Suggesting Friends using the Implicit Social Graph. In *Proc. KDD 2010*, ACM Press (2010), 233-242.

35. Secord, A., Winnemoeller, H., Li, W. and Dontcheva, M. Creating Collections with Automatic Suggestions and Example-Based Refinement. In *Proc. UIST 2010*, ACM Press (2010), 249-258.

36. Settles, B. Closing the Loop: Fast, Interactive Semi-Supervised Annotation with Queries on Features and Instances. In *Proc. EMNLP 2011*, (2011), 1467-1478.

37. Sheehan, K. Towards a Typology of Internet Users and Online Privacy Concerns. *The Information Society 18*, (2002), 21-23.

38. Siegler, M. Zuckerberg: "Guess What? Nobody Wants to Make Lists." Aug. 26, 2010. http://techcrunch.com/2010/08/26/facebook-friend-lists/.

39. Slee, M. Friend Lists. Dec. 19, 2007. http://www.facebook.com/blog.php?post=7831767130.

40. Smetters, D.K. and Good, N. How Users Use Access Control. In *Proc. SOUPS 2009*, ACM Press (2009).

41. Toomim, M., Zhang, X., Fogarty, J. and Landay, J. Access Control by Testing for Shared Knowledge. In *Proc. CHI 2008*, ACM Press (2008), 193-196.

42. van Zwol, R., Sigurbjörnsson, B., Adapala, R., Pueyo, L.G., Katiyar, A., Kurapati, K., Muralidharan, M., Muthu, S., Murdock, V., Ng, P., Ramani, A., Sahai, A., Sathish, S.T., Vasudev, H. and Vuyyuru. U. Faceted Exploration of Image Search Results. In *Proc. WWW 2010*, ACM Press (2010), 961-970.

43. Whalen, T., Smetters, D., and Churchill, E.F. User Experiences with Sharing and Access Control. *Ext Abstracts CHI 2006*, ACM Press (2006).

44. Wobbrock, J.O., Findlater, L., Gergle, D. and Higgins, J.J. The Aligned Rank Transform for Nonparametric Factorial Analyses using only ANOVA Procedures. In *Proc. CHI 2011*, ACM Press (2011), 143-146.

45. Wong, W. Oberst,I., Das,S., Moore, T., Stumpf, S. McIntosh, K. and Burnett, M. End-User Feature Labeling: A Locally-Weighted Regression Approach. In *Proc. IUI 2011*, ACM Press (2011), 115-124.

46. Yelp.com. http://www.yelp.com