# Group Proximity Measure for Recommending Groups in Online Social Networks

Barna Saha
Department of Computer Science
University of Maryland College Park
College Park, MD 20742
barna@cs.umd.edu

Lise Getoor
Department of Computer Science
University of Maryland College Park
College Park, MD 20742
getoor@cs.umd.edu

## Abstract

*There are currently, thousands of online social networks (OSN) available, each of which hosts millions of users. Users make new friendship links, join groups on diverse topics, share opinions and thus help in building a big knowledge repository. In this paper we study the problem of defining proximity measure between groups (communities) of OSN. Understanding the proximity among the groups in OSN can reveal new insights into the structure of the network. We use this proximity measure in a novel way with other structural features of online groups to predict the evolving state of the network. One of the important task in the sphere of OSN is helping users in selecting groups of their interest, by making recommendations. We utilize group proximity measure to propose a new framework for recommending groups to users of OSN. Finally we investigate the effectiveness of our methods in three real OSN: Flickr, Live Journal and You Tube. Our methods give promising results.*

## 1. Introduction

Social networking is becoming a ubiquitous feature of our online life. Social networking sites are attracting one out of every 20 Web visits [14]. In the month of September 2006, one out of every 20 U.S. Internet visits landed on one of the top 20 social networking Web sites, nearly double the share of visits compared with a year ago. A few of the most popular online social networks are Facebook, MySpace, LinkedIn, Reunion.com, Flickr, Live Journal, Orkut, Shefali, AOL.com etc. These networks give users a platform to link with similar minded people, join groups and share their views on topics that compel their interest. As an example at present Orkut has 6,7000,000 users and 4,7092,584 groups. The size of the other networks are equally huge. This increasing gain in popularity of OSN has resulted in surge of interest in modeling and understanding the social interactions and dynamics of these networks [1, 6, 5].


The 2nd SNA-KDD Workshop '08 ( SNA-KDD'08), August 24, 2008 , Las Vegas, Nevada , USA.


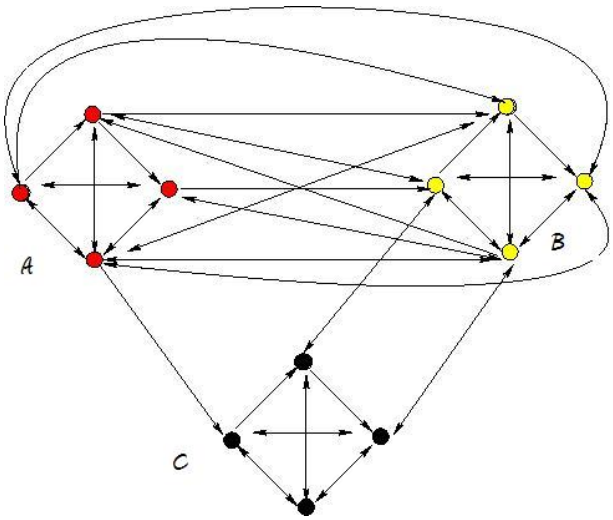


**Figure 1. Simple example of group proximity . Nodes are members of an OSN. Edges represent friendship links. There are three groups $A, B, C$. Members of each group are colored differently.**

One of the most important features and underlying reasons for the popularity of OSN is that they allow users to create groups. Users can start groups on different topics. Users can join the groups already created by someone else. They can initiate discussions, write posts to share their opinions, ideas and experiences.

While there has been a significant amount of research in describing node proximity, there has been little research in designing measures for group proximity. This may be due to the ambiguity of semantics of closeness between two groups. Here we characterize group proximity in two orthogonal ways. One is *topic-based proximity*, where two groups are said to be close if they have similar topics of discussion. The other one, which we pursue in this paper is *link-based proximity*. It assigns group $B$ as close to group $A$, if

a member of group $A$ can easily reach group $B$ using friendship links. Figure 1 shows a simple example of an undirected network. From the figure it is clear that group $A$(colored red) and group $B$(colored blue) are much closer to each other than group $A$ and $C$(colored green). However in the example, each group is a clique and disjoint. This does not quite reflect the structure of groups as observed in real-world networks. In real networks members participate in numerous groups. For example, in Orkut the average number of groups in which a user belongs is around $50 - 100$. So the groups have overlapping memberships.

In addition it is not possible for a member who belongs to many groups to participate equally in all the group activities. Within a group generally, there is a set of members who are closely knit and actively guide the discussions. The other set of members are silent observers and their participation in group activities is much less. Thus not all the members in a group are equally important from the perspective of the group (structurally and functionally). Orkut is an undirected network, but most of the other real networks like Flickr, Live Journal, You Tube are directed. Our approach can handle both undirected and directed networks and is built upon Tong et al's work of [19]. The *group* as defined by them is any collection of nodes and is not necessarily a community. Thus their measure of group proximity does not consider the salient features of groups in OSN. In Section 2, we describe how their measure can be modified suitably for defining group proximity measure in OSN.

While understanding the relationship between groups is itself an important ingredient in the analysis of online social networks, they can be used in varieties of other applications concerning OSN. Two important tasks, where we employ the metric of group proximity are predicting growth of the network and for recommending groups in OSN.

### 1.1 Predicting Growth of Networks

Given the current status of a network, we are interested in answering the question "which new members will join a group in near future ?". Most of the people are influenced by the opinions of their friends. This is the basis for the "word of mouth" effect and the works on information diffusion, product recommendations all rely on it. Thus it is likely that those users who develop friendships with members of a given group, are likely to join that group. Predicting growth of a group is hence directly related to the problem of *link-prediction* [12]. If we can predict members of which groups are likely to develop strong friendship among themselves in near future, we can almost surely conclude that these groups will share a large common membership shortly. Their discussion themes will follow the same trend. In Section 3, we show how group proximity measure can be utilized for this.

### 1.2 Recommending Groups in OSN

Voluminous number of groups in OSN makes it impossible for a user to successfully search groups, matching his interest. Solely searching for the groups in which his friends are members, might require browsing hundreds of groups. A recommendation system is useful in this scenario, for finding groups which a user will like to join.

While recommendation systems have received ample of attention in the domains of movie recommendation, product recommendation, document recommendation etc., not much work has been put for recommending groups in OSN. As far as we know the work of Isabelle et al [18] is the only one to consider group recommendations in social networks. However their prediction accuracy is never above $48\%$ and that too using a very mild success condition. In section 4 we propose a new approach of group recommendations in OSN utilizing the group proximity and the group membership history.

### 1.3 Contribution

- We develop new link-based proximity measure for groups in online social networks (Section 2).
- We show how to use group proximity measure along with other structural properties of groups for predicting number of new links that will develop between any two groups. (Section 3)
- We propose a new recommendation system based on group proximity and the history of user's group membership. (Section 4)
- We perform a detailed evaluation on a sample of Flickr, Live Journal and You Tube datasets to investigate the effectiveness of our methods. (Section 5)

## 2. Group Proximity Measure

In this section, we describe our group proximity measure and the algorithm for computing it. We make use of the concept of escape probability from [19]. The work there proposed a measure of proximity between a collection of nodes, by first picking a random node $v$ from the collection, say $X_1$, and then estimating the probability that a random walk starting from $v$ visits a node in the other collection of vertices, say $X_2$, before visiting any node in $X_1$.

The above approach has a number of drawbacks, if directly applied to group proximity measure in OSN. First, in OSN as described in the introduction, all the nodes in a group are not equally representative of the group. The rolls each member plays in a group are different. The members which have been in the group for a long time and have participated in active conversations are more reliable members of the group than those who have just joined the group and have low level of activities. Selecting a node for initiation of the random walk, without considering the roll the corresponding member plays can not distinguish between these two categories of members in a group. It will perform much inferior to an approach which gives due importance/weight to the actual representatives of the group.

Secondly, in the groups of OSN, the representative members form the core of the group, with many links among themselves. A random walk that starts from such a node is likely to hit a member of the same group with high probability. Thus the simple escape probability based measure, starting from a representative member assigns very low value of proximity to all the remaining groups and cannot not differentiate among them. On the otherhand the other members of a group play the roll of outliers and generally have very few links going inside the group. Random walk that initiates from these corresponding nodes will visit other groups, but measuring proximity between two groups based on these nonrepresentative members is not reliable.

## 2.1 Preliminaries

Before, we describe our approach, we first introduce some concepts based on random walks on undirected and directed graphs. A graph $G$ has a set of vertices $V$, numbered 1 to $n$ and a set of edges $E = \{(i,j)|i,j \in V\}$. The adjacency matrix $A$ of $G$ is a $n \times n$ matrix, where the entry in the $(i,j)$th cell, $A_{i,j}$ gives the weight of the edge from $i$ to $j$. If there is no edge from $i$ to $j$ the entry is 0. When the graph is undirected $A$ is symmetric. $D$ is a diagonal matrix, where $D_{i,i}$ represents the degree of $i$th vertex in the case the graph is undirected and the number of outgoing edges from $i$, when the graph is directed. So $D_{i,i} = \sum_{j \in V} A_{i,j}$.

Consider a random walk that is at node $v_t \in G$ at time $t$. Then if $v_t$ has $N(v_t)$ neighbors, at timestep $t+1$, the random walk visits the node $w \in N(v_t)$ with probability $\frac{A_{v_t,w}}{\sum_{x \in N(v_t)} A_{v_t,x}}$. The sequence of nodes $v_1, v_2, .., v_t$ thus forms a Markov chain. Denote by $P_t(i) = Prob(v_t = i)$. $P = \{p_{i,j}\}$ represents the transition matrix of the Markov chain, where $p_{i,j} = \frac{A_{i,j}}{D_{i,i}}$ if $(i,j) \in E$, otherwise it is 0.

*Escape Probability:* The escape probability $E_{i,j}$ from node $i$ to node $j$ is the probability that a random walk starting from node $i$ will visit node $j$ first, before visiting node $i$. Recursively it can be written as

$$E_{i,j} = \sum_{k=1}^{n} p_{i,k} v_k(i,j) \quad (1)$$

where $v_k(i,j)$ denotes the probability a random walk starting from node $k$ will visit node $j$ before visiting node $i$. So we have the following linear system:

$$v_k(i,j) = \sum_{t=1}^{n} p_{k,t} v_t(i,j) \text{ if } k \neq i,j$$
$$v_i(i,j) = 0, v_j(i,j) = 1 \text{ otherwise} \quad (2)$$

Solving the above linear system we get, $E_{i,j} = P(i,\mathcal{I})GP(\mathcal{I},j)$, where $\mathcal{I} = V - \{i,j\}$, $P(i,\mathcal{I})$ represents the $i$th row of $P$ without $i$th and $j$th element and $P(\mathcal{I},j)$ represents the $j$th column of $P$ without $i$th and $j$th element. $G = (\mathbf{I} - P(\mathcal{I},\mathcal{I}))^{-}$. Here $\mathbf{I}$ is the identity matrix and $P(\mathcal{I},\mathcal{I})$ represents matrix $P$ after removing the $i$th and $j$th row. We know matrix inversion takes $O(n^3)$ time. So if this method is directly applied to compute all pair escape probabilities, the time complexity will become $O(n^5)$. Tong et al's [19] proposed a very efficient algorithm, which takes only $O(n^3)$ time to compute all pair proximities in a network.

## 2.2 Our approach

We extend the above idea for group proximity in two ways. First we are able to assign importance to the members in a group, who are representatives of the group. Second our approach creates a concise graph which is much smaller in size and thus improves the time-complexity. Below we describe the major steps of our algorithm and elaborate them in detail in the coming subsections.

We begin by dividing the members in each group into two partitions: core and outlier. Members in the core of a group are the representatives of the group. They have many connections inside the group and actively participate in the group activities and influence others to join the group. The members in the outliers have very few links in the group and may have just joined the group. Let $G_i$ and $G_j$ be two groups. Let $C_i(C_j)$ represents the core of $G_i(G_j)$ and $O_i(O_j)$ represents the outliers of $G_i(G_j)$. Our algorithm has three main steps:

1. **Find CORE:** Find $C_i$ and $C_j$ of $G_i$ and $G_j$ respectively.
2. **Obtain Concise Graph:** Shrink $C_i$ and $C_j$ into two vertices $V_i$ and $V_j$ respectively. Remove the self loop. Replace the parallel edges by a single edge with proper weight, taking care of the common memberships. Denote the smaller graph by $G'$.
3. **Compute Escape Probability:** Compute $E_{V_i,V_j}$ in $G'$.

Note that here no random choice of node is required for escape probability computation. Because of shrinking, the disadvantage of a random walk starting from a representative member hitting the core of the same group is eliminated. The shrunken graph formation takes care of the different rolls of the core and the outliers. The next two subsections describe the first two steps in details. Once the concise graph is formed, the third step is performed using the fast algorithm of [19].

## 2.3 Finding CORE of a Group

We use two approaches for finding the *core* of a group. The first approach uses simple degree centrality. For every node in a group, compute its induced degree inside the group, that is the number of members of the group the node is linked to. The algorithm returns all the nodes whose induced degree is $\geq \frac{1}{\alpha}$ of the total size of the group, where $\alpha$ is a constant. Here we rely on the assumption that core members of a group have many friends inside the group. Instead of degree, we can also pick the subgraph within a group, which has maximum ratio of edges/vertices. We used this simple measure in our experiments and got reasonable results.

The second approach uses truncated commute time [13] inside each group. The commute time between node $x$ and $y$ is defined as the total time required by a random walk starting from node $x$ to hit node $y$ and then to come back to $x$. The core of a group forms a closely knit component. So commute time between any two members of the core should be smaller. We use a greedy method for finding the core. Let $C_i$ denote the core of the group $G_i(V_i, E_i)$ and $M$ denote the all pair commute time matrix for $G_i$. The following algorithm computes the core $C_i$ of the group $G_i$. It starts by finding the two nodes, which have minimum commute time and includes them in the core. At every iteration, it finds the member not included in the core so far, which has minimum average commute time to the members of the core. If that time is less than some chosen threshold, the member is inserted in the core and the process is repeated.

**Algorithm 2.1:** CORECOMMUTETIME($G_i$)

Find $(x,y)$ with minimum $M(x,y)$.
$C_i \leftarrow \{x,y\}$
$MIN \leftarrow M(x,y)$
**while** $MIN < Threshold$
  **do** Find $k$ with minimum $\sum_{a \in C_i} M(k,a)$
$C_i \leftarrow C_i \cup \{k\}$
$MIN = \frac{\sum_{a \in C_i} M(k,a)}{|C_i|}$
**output** $(C_i)$

### 2.4 Obtaining Concise Graph

Once the core $C_i$ of group $G_i$ has been identified for all the groups. We shrink $C_i$ into a single vertex. The algorithm below describes the shrinking procedure. The algorithm takes as its input all the cores and the corresponding groups along with the social network graph.

**Algorithm 2.2:** CONCISEGRAPH($\{G_i, C_i\}, G$)

**for each** $(i,j) \in E(G)$
**do** {
  **if** $i \in C_x and j \in C_y \forall x, y$
    **then** $A_{G'}(V_i, V_j) += 1$
  **if** $i \in C_x and j \in G_y - C_y \forall x, y$
    **then** $A_{G'}(V_i, j) += 1$
  **if** $i \in G_x - C_x and j \in G_y - C_y \forall x, y$
    **then** $A_{G'}(i, j) += 1$
**for each** $i \in V(G)$
**do** {
  **if** $i \in C_x and i \in C_y \forall x, y$
    **then** $A_{G'}(V_i, V_j) += \alpha$
  **if** $i \in C_x and i \in G_y - C_y \forall x, y$
    **then** $A_{G'}(V_i, V_j) += \beta$
  **if** $i \in G_x - C_x and i \in C_y \forall x, y$
    **then** $A_{G'}(V_i, V_j) += \gamma$
REMOVE all the vertices whose degree becomes 0 in $G'$
**output** $(G')$

The first for loop creates the shrunken graph. The second for loop assigns weight to the edges based on whether two groups share a core member, whether a core member of a group is an outlier of a group and whether two groups have a common outlier member. This weight assignment captures variable degree of sharing among groups. For every common member $x$ of $G_i$ and $G_j$, if $x \in C_i \cap C_j$, then the weight of the edge $(V_i, V_j)$ increases by $\alpha$. If $x \in C_i \cap O_j$ or $x \in O_i \cap O_j$, the weight of $(V_i, V_j)$ increases by $\beta$ and $\gamma$ respectively. Here $\alpha > \beta > \gamma$. In the experiment using Flickr, this procedure was able to reduce the number of vertices to half, and edges to $\frac{1}{10}$th fraction of the total edges. Thus the advantage gained in running time is quite clear, since the time for computing escape probability dominates the time complexity for creating the shrunken graph.

## 3. Predicting Future Growth of Groups

As discussed in the introduction, OSN typically grow over time. New users join the network, new groups get created. Users make new friends, participate in new groups. It is a challenge to analyze the huge dynamics of such networks. The foundation of such an analysis rests on modeling the evolution of the network. If we can predict from the current state of the network how it will grow in near future, we can use that knowledge to build efficient algorithms to handle the temporal evolution of the network.

We use our group proximity measure to predict how many new links will develop between any pair of groups in near future. If this can be predicted with reasonable accuracy, it will enable us to discover members of which two groups will develop friendships. It is likely that members of one group will also join the other group in near time.

### 3.1 Link Cardinality Estimation

For every pair of groups, we use the group proximity value between these groups, the current number of links and the product of the sizes of each group as the features, upon which to base our predictions. Using the values of these features at time $t$, we would like to determine how many links will there be between any two groups in recent future at time $t + \delta$. Each of these features are normalized appropriately. We select a sample of the data for training. For them, we know the proximity values at $t$ and the number of links at both $t$ and $t + \delta$. For the test data we do not know the number of links at time $t + \delta$. We learnt a regression tree on the training samples. Regression tree is a variation of decision tree where the class labels can be any real numbers. We used some other learning techniques as well. The details of the methods are described in the Experimental section.

It is to be noted that in a similar way, we can predict for a particular node and group, how many friends a node will develop in the group. We can use our group proximity measure, considering the particular node as a singleton group and then learn a regression tree based on the proximity value and size of the said group.

## 4. Group Recommendation in Online Social Networks

In this section we show how we use our proximity measure for recommending groups in an OSN. Collaborative filtering [7] is one of the popular techniques used in different recommendation systems. It tries to identify members with similar taste (like same kind of movies, same kind of pruducts) to a user and recommend based on what those members have liked. The recommendation system [18] uses a similar kind of collaborative filtering approach for recommending groups where it is assumed a user likes to join a group where his or her friends have joined. This we refer as *user level recommendation*.

Our approach is also based on collaborative filtering. However instead of considering only friendship links to identify like-minded people, we take into account the current group membership information of a user. Members join a group, because they share common interest. We call our recommendation system, which considers the collective behavior of both user's friends and the members of the groups in which the user currently belongs the *group-level recommendation*. Precisely our recommendation system is based on the following assumptions:

- A user joins a group, because s/he reaches that group through his or her friendship links and the groups s/he is already a member of.
- Groups which are far apart in the network from the groups in which user is currently a member, are unlikely to match user's interest.

The first assumption is based on user's browsing pattern. As a user browses through the network, s/he is likely to visit his or her close neighbors (friends, friends of friends etc) in the network through friendship links and those members who belong to the groups s/he is a member of, their friends and so on. In turn s/he gets influenced by the groups these users belong to and select to join one such group. The second assumption is based on the fact that a group which is not easily reachable through friendship links from the groups in which a user currently belongs is unlikely to

match the interest of the user. No other user with same kind of interest has joined that far away group.

To understand better the strength of the group level recommendation system and why group proximity is useful, consider two groups, such that a large fraction of members in the first group have many friends in the second group. Since the members of the same group naturally have common interest, the fact that a good fraction of like-minded people have many friends in the second group serves as a support that the users in the first group will like to join the second group. Thus this is not based on the friendship links of a single user, but depends on the distribution of friendship links of a big collection of users. Our algorithm for group proximity will assign high values to these pair of groups. Group proximity is also affected when two groups share common members. Presence of a significant number of common members between two groups is a strong evidence that users in one group joined the other group and vice versa. Therefore any user of the first group, who is not a member of the second group will probably like to join the second group.

Below we describe the exact methodology for group recommendation.

## 4.1 Group Recommendation Method 1

This is a simple deterministic method. Here for each user we want to recommend a group. We utilize the information of the current group membership and the group proximity. Let user $A$ belong to groups $G_A = G_1, G_2, \ldots, G_l$. Suppose $\mathcal{G}$ represents the entire collection of groups. Then for every group $G \in \mathcal{G} - G_A$, we compute the total proximity of these groups from the groups in $G_A$. We use the computed value as the score for every group in $\mathcal{G} - G_A$ for user $A$. We recommend the $k$ groups with top scores to user $A$. Below is the description of the algorithm. In the algorithm $Prox(h, g)$ represents the proximity value of $G$ from $H$.

**Algorithm 4.1:** GROUP-RECOMMENDATION1$(A, G_A, \mathcal{G}, Prox)$

**for each** $G \in \mathcal{G} - G_a$
  **do** $\begin{cases} score(G) \leftarrow 0 \\ \textbf{for each } H \in G_A \\ \quad \textbf{do } score(G) \leftarrow score(G) + Prox(H, G) \end{cases}$
Select the top-$k$ groups by $score$
Return the selected groups as the recommended ones

## 4.2 Group Recommendation Method 2

Method 1 assigns all the current groups equal importance in determining the next group user will join. However in many cases, a user does not participate in all his groups equally. In method 1 there is no way of finding which groups influence a user in choosing the next group. To capture this we propose the concept of group closure and employ a new classifier over it.

*Group Closure.* For each group, we define its closure as all the groups close to it by the proximity measure. In general, while computing the closure of $G_i$, any group, whose proximity is higher than the average proximity of all the groups from $G_i$ is included in the closure of $G_i$. Algorithm 4.2 gives the procedure for finding the closure of $G_i$.

Note that these closures can overlap with each other. A different approach for finding closure will be to use clustering techniques. We can define a complete graph where there is a node for every group. An edge from $G_i$ to $G_j$ has weight $Prox(G_i, G_j)$. Note that the edge weights are neither symmetric nor they follow triangle inequality. We are currently investigating what kind of graph clustering techniques will be useful to identify the closures.

**Algorithm 4.2:** GROUP-CLOSURE$(G_i, Prox, \mathcal{G})$

$Total \leftarrow 0$
**for each** $G \in \mathcal{G} - G_i$
  **do** $Total \leftarrow Total + Prox(G_i, G)$
$GC_i \leftarrow \{G_i\}$
**for each** $G \in \mathcal{G} - G_i$
  **do** $\begin{cases} \textbf{if } Prox(G_i, G) \geq \frac{Total}{|\mathcal{G}|-1} \\ \quad \textbf{then } GC_i \leftarrow GC_i \cup \{G\} \end{cases}$
**return** $(G)C_i$

Group closure helps in identifying the groups which play the most significant roll in determining the next group for a user. The assumption is that if a user belongs to many groups in the closure of say $G_i$, then $G_i$ is one of the key groups that guides the user in his or her next choice. We define three categories of memberships of a user in each closure: high, medium and low. If a user belongs to $\geq \delta_1$ fraction of the groups in a closure, s/he is a high member of the closure. If the user is a member of $< \delta_1$ but $\geq \delta_2$ fraction of the groups in a closure, s/he is a medium member. Otherwise s/he is said to be a low member of the closure.

*Classifier Description.* We assume there exists a fraction of users, whose group membership information is available entirely. That is for them, we know the latest group they joined and also the rest of the membership history. This serves as the training data for the classifier which we will describe next. For the remaining users we do not know the last group the user became a member of. Rest of the membership history is available. Using this available membership history and the learnt classifier, we want to predict the latest group which these users joined.

From the training data available, we learn the probability distribution that given the category of memberships of a user in all the closures, the probability that s/he joins a particular closure next. Here we assume full independence. That is

$$Prob(\text{ user } A \text{ joins } GC_i \big| m(u, GC_j) = a_j \forall j = 1 \text{ to } |\mathcal{G}|)$$
$$= \prod_{j=1}^{|\mathcal{G}|} Prob(\text{ user } A \text{ joins } GC_i \big| m(u, GC_j) = a_j)$$

Here $m(A, GC_j)$ denotes the category of membership of user $A$ in closure $GC_j$. $a_j$ for all $j$ can take value among high, medium and low. We also learn the probability distribution of users joining a group, conditioned on their joining a particular closure. These distributions are learnt from the training data sets. For every user $A$ for whom we want to recommend groups, we compute a score

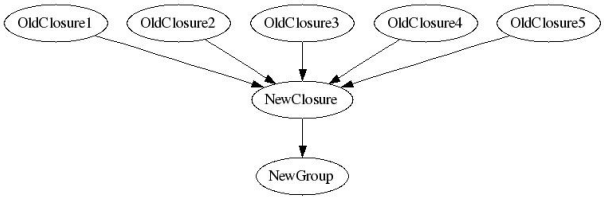


**Figure 2. Classifier for group recommendation**

for every group in $\mathcal{G} - G_A$, using the above two distributions.

$$Score(G \in \mathcal{G} - G_A)$$
$$= \sum_{i=1}^{\mathcal{G}} Prob(\text{ user } A \text{ joins } G \big| \text{ user } A \text{ joins } GC_i) *$$
$$Prob(\text{ user } A \text{ joins } GC_i \big| m(A, GC_k) = a_k \forall k = 1 \text{ to } |\mathcal{G}|)$$
$$= \sum_{i=1}^{\mathcal{G}} Prob(\text{ user } A \text{ joins } G \big| \text{ user } A \text{ joins } GC_i) *$$
$$\prod_{k=1}^{|\mathcal{G}|} Prob(\text{ user } A \text{ joins } GC_i \big| m(u, GC_k) = a_k)$$

The Figure 2 depicts the structure of the classifier. The first level contains a random variable corresponding to the category of user's membership in each closure. The second level contains a single variable for the new closure user joins. The third level contains the variable for the new group user joins.

The assumption of full independence here is a bit over-simplified, as the closures may have significant overlaps. However as we show in the experimental section, even with this strong assumptions of independence, the method 2 performs quite well.

# 5. Experimental Results

In this section, we describe our experimental findings on three different real world online social networks: Flickr, You Tube and Live Journal. The datasets are collected from http://socialnetworks.mpi-sws.mpg.de/data-imc2007.html. We selected a random sample from each of these three networks for our experimental purpose. The size of the samples are given in Table 1.

| Network | # of members | # of links | # of groups |
|---|---|---|---|
| Flickr | 2000 | 1,40,7292 | 100 |
| You Tube | 2000 | 32,881 | 100 |
| Live Journal | 2000 | 48,165, | 100 |

**Table 1. Data Description**

Each sampled datasets contain friendship links and group membership information. We ran several experiments on the datasets, using different settings and classifiers. We used LIBSVM [3], DTREG [17] and WEKA [9] for SVM classifier, regression tree and linear regression respectively. Rest of the codes are written by us. We first illustrate the methodology and results for predicting link cardinality between groups and then show the outcomes of our group recommendation system.

| Classifier | Measurement Parameter | Flickr |
|---|---|---|
| Regression Tree | Mean Absolute Error | 21.84% |
| Linear Regression | Mean Absolute Error | 30.8422% |
| SVM-Multiclass | Witihin Label+/-1 | 64% Success |
| SVM-Multiclass | Within Label+/-2 | 81% Success |
| Classifier | Measurement Parameter | You Tube |
| Regression Tree | Mean Absolute Error | 18.62% |
| Linear Regression | Mean Absolute Error | 28.54% |
| SVM-Multiclass | Witihin Label+/-1 | 79% Success |
| SVM-Multiclass | Within Label+/-2 | 88% Success |
| Classifier | Measurement Parameter | Live Journal |
| Regression Tree | Mean Absolute Error | 22.54% |
| Linear Regression | Mean Absolute Error | 26.73% |
| SVM-Multiclass | Witihin Label+/-1 | 68% Success |
| SVM-Multiclass | Within Label+/-2 | 84% Success |

**Table 2. Results of link cardinality estimation on Flickr, You Tube and Live Journal using regression tree, linear regression and a multiclass SVM classifier**

## 5.1 Results of Link Cardinality Estimation

For link cardinality estimation, for each pair of groups we used group proximity measure, current number of links in between and the normalized product of the size of the two groups as features. We sampled edges uniformly and randomly with probability $1/2$. The sampled edges form a network with the number of edges approximately half of the actual network. All the features are computed based on the sampled network. Using these features, we want to predict the link cardinality between any two groups in the original network. We used 10 fold cross-validation and used 3 different kinds of methods. Two of these methods are based on regression: linear regression and regression tree. As expected using regression tree outperforms linear regression. Using a regression tree we get an average absolute error around 22% for Flickr, which indicates if the true value is $X$, then the predicted value is within $X(1 \pm 0.22)$. For using a classifier, we first discretize the number of links for both current and previous, into nonoverlapping ranges of 100. For example, if the number of links is between 0 to 100, it is given a class label 1. If the number of links is between 101 to 200, it is given a class label 2 and so on. Using a SVM-multiclass classifier, we get for 64% pairs of groups in Flickr, if the actual number of links is $X$, then the predicted number is within $X \pm 100$. The entire results are tabulated in the Table 2. Note that the first two rows tabulate the mean absolute error for Flickr, You Tube and Live Journal. So the lower are these values, better is the result of regression. The values in the last two rows on the other hand indicate for what percentage of the group-pairs the classifier was able to predict the number of links very close to the original. In this case, the higher these values are, the better is the result.

## 5.2 Results of Group Recommendation

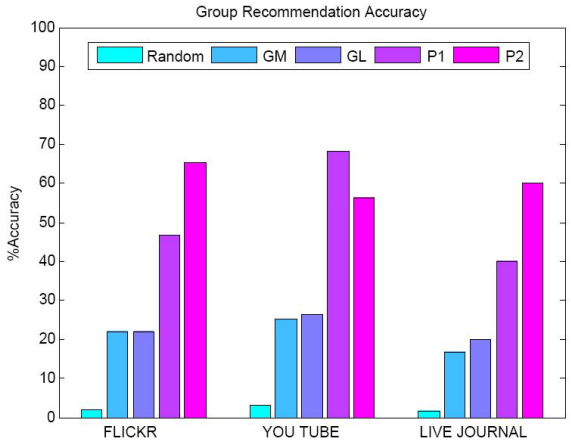


**Figure 3. Comparison of group recommendation results for Flickr, You Tube and Live Journal using random group selection, GM, GL, method 1 (P1), method 2 (P2)**

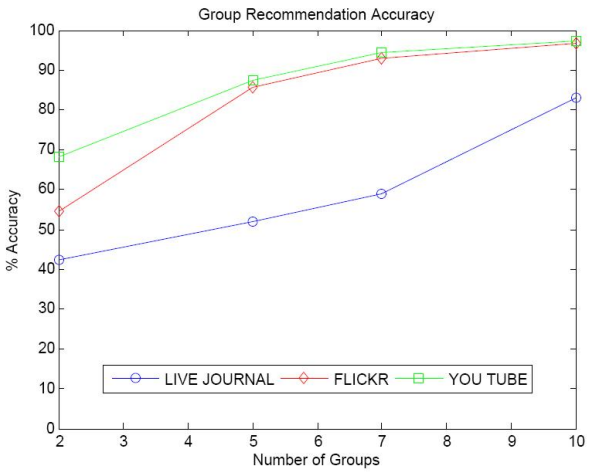


**Figure 4. Change in group recommendation accuracy with increasing membership on Flickr, You Tube and Live Journal with milder success condition**

For group recommendation we show empirical results for both method 1 and 2 as described in the previous section and compare them with some other classifiers. For both method 1 and method 2, we return the top 5 groups based on the score they assign and check whether the group to be predicted is one of them. For experimental purpose, we first tried to learn SVM multiclassifiers using LIBSVM [3]. Group-Only SVM-Multiclass (GM), uses the binary vector of group membership as feature (excluding one group) and tries to predict the left out group which may be any of the 100 groups for every user. Total-Link SVM-Multiclass (GL), learns a SVM multiclassifier, where the feature vector consists of the number of links each new group has to the current user's groups. Similar to GM, here also the classifier tries to predict the left out group for every user. Using 10 fold cross-validation we get an accuracy of around 22% for both the cases for Flickr, which is not very good. The result using these classifiers is also not promising for Live Journal or You Tube. Using method 1, our accuracy was raised to 46.65% for Flickr, that is for 46.65% of the users method 1 was able to predict the latest group accurately. Method 1 performs exceptionally well for You Tube, where the accuracy is above 60%. Method 1 does not perform as well for Live Journal, where we are able to predict successfully in around 40% of the cases. Method 2 uses our proposed classifier and 10 fold cross validation. The accuracy is above 65% for Flickr and more than 55% for the other two. Figure 4 gives the comparison result among the methods. The first bar in the figure 4 represents the accuracy when a random group is chosen for each user and matched with the group to be predicted. As expected for random choice the accuracy is very low.

So far, we have considered our prediction successful if the one particular group to be predicted belongs to the 5 groups that are returned by method 1 or 2. We now use a milder success condition as described in the work of Isabelle et al [18]. For each user in the test sample, which is small compared to the actual number of users, they obtained a list of 4 groups for Live Journal and a list of 20 groups for Orkut. If for a user at least one common group exists between the entire list of groups the user is a member of and the predicted list of groups for the user, it is counted as success. They achieved a 48% success in both the networks using this mild success condition. Figure 5 shows the group recommendation accuracy, if we take a weaker success criteria. We leave aside half the groups from each user's membership vector. Considering only the remaining groups and using method 1, we then find out if there is a common group between the predicted list of 5 groups and the groups that are left out. The accuracy is raised to above 80% for both Flickr and You Tube, when we use this success measure for method 1 and consider only those users who belong to at least 5 groups. The accuracy for Live Journal is still low compared to Flickr and You Tube. But even for Live Journal it is more than 55% when users belong to at least 5 groups and above 80% when they belong to at least 10 groups. Figure 5 shows how the accuracy increases as we consider only the users who belong to at least a certain number of groups. In the figure if the accuracy is $a\%$ when the x-coordinate is $b$, then for $a\%$ of those users who belong to at least $b$ groups, the method predicted successfully.

The results clearly show that using group membership or link cardinality alone is not sufficient for group recommendation. But when group proximity is coupled with group membership the recommendation system becomes very effective. Method 2 outperforms method 1 in both Flickr and Live Journal. In You Tube method 1 performs better. When using a weaker condition for success our method performs extremely well and gives result much better than [18].

## 6. Related Work

In this section we briefly describe some of the works done on social networks related to our work.

There has been large amount of research in defining proper measure of node proximity in social networks. As we know many

of the real networks have the scale-free property and exhibit the small world phenomenon. Any two nodes in such a network generally have a short path connecting them. Thus a short path length does not necessarily capture the friendship bond or the closeness (proximity) between two nodes. Faloutsos et al [4] modeled the proximity between two nodes by introducing the notion of connection subgraphs. Connection subgraphs use the concept of current flow in electrical network and return a small subgraph $H$ which demonstrates the relationship between a given set of query nodes. The work of Ramakrisnan [15] attempted to find connection subgraphs from multirelational data. These works are based on the assumption that the underlying network is symmetric. Tong et al [19] proposes asymmetric proximity measures in directed graph using the techniques of random walk and escape probability. Random walk has been widely used for several tasks including computing pagerank [11], finding expected node distance (random walk with hitting and commute time)[13], in semisupervised and supervised learning [20].

OSN grows rapidly. One of the crucial part of network analysis in OSN is to understand the dynamics of it. Predicting whether a link will be created between two nodes (link prediction) [12], modelling how information spreads (information diffusion) [10], and groups are formed [2] are some of the works done in this area. Recommendation systems have received significant attention for recommending movies, products, research papers etc [7, 16]. The major techniques can be broadly classified into two categories content based filtering and collaborative filtering. Content based filtering is developed on the premise that user will like a product, if he has liked similar kind of products in the past. Whereas collaborative filtering tries to identify other users with taste similar to the current users and recommend the products which those users have liked. There are some approaches which try to merge both of these. Here we are interested in recommending groups to users in OSN.

There is a clear distinction between the recommendation systems for OSN from the rest. For example, in the case of movie recommendation, if a user enjoys thrillers, he almost surely will enjoy the high quality movies in that genre. However in OSN, there are groups on varieties of topics with different flavors. It is very unlikely that a user will join many groups on similar topic. So content based filtering will not suit for OSN. The work of Isabelle et al [18] is one of the first to consider group recommendation in social networks. They consider the model of collaborative filtering [8]. They explicitly discover the clusters in the network based on friendship links and find the probability distribution of groups in each cluster. They use this information to recommend groups to users who has just joined the network and whose friendship links are only known. However their recommendation is solely for the users who are new in the network and they do not consider a continuous recommendation system, which recommends groups to both veterans and newcomers. Since their recommendation system is tipped towards newcomers, they do not use the current membership information of a user to recommend a new group. When they predict 4 groups for LiveJournal and 20 groups for Orkut, in 48% users they are successful in predicting atleast one group among all the groups in which the user is a member of. We are not sure, how their success rate will be affected if they try to predict the last group which user joins. It is very likely that the success rate will drop.

# 7. Future Work and Conclusion

We have introduced a new measure of group proximity in OSN using friendship links across groups and common memberships. We use group proximity along with some other structural features of groups to estimate link cardinality between groups. Accurate estimation of link cardinality helps in understanding the evolution of groups. We then propose two methods for recommending groups in OSN using group proximity and membership history. The experimental results on Flickr, Live Journal and You Tube validate the effectiveness of our measure.

Our proximity measure does not use any attribute information of groups. Incorporating attributes which are common among majority of members of the groups may improve the proximity measure. Our group recommendation system is based on the assumption that members in a group share similar interest. But there exists some very generic large group like USA. The assumption that members of a group is like-minded does not hold here. We are currently developing a Bayesian network framework to model this aspect.

# 8. References


[1] Yong-Yeol Ahn, Seungyeop Han, Haewoon Kwak, Sue Moon, and Hawoong Jeong. Analysis of topological characteristics of huge online social networking services. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 835–844, 2007.

[2] Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. Group formation in large social networks: membership, growth, and evolution. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 44–54, 2006.

[3] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[4] Christos Faloutsos, Kevin S. McCurley, and Andrew Tomkins. Fast discovery of connection subgraphs. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 118–127, 2004.

[5] Ralph Gross, Alessandro Acquisti, and III H. John Heinz. Information revelation and privacy in online social networks. In *WPES '05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 71–80, 2005.

[6] Jeffrey Heer and Danah Boyd. Vizster: Visualizing online social networks. *infovis*, 0:5, 2005.

[7] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.

[8] Thomas Hofmann and Jan Puzicha. Latent class models for collaborative filtering. In *IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 688–693, 1999.

[9] G. Holmes, A. Donkin, and I.H. Witten. Weka: A machine learning workbench. In *Proc Second Australia and New*

*Zealand Conference on Intelligent Information Systems*, Brisbane, Australia, 1994.

[10] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146, 2003.

[11] Amy Langville and Carl Meyer. Google's pagerank and beyond: The science of search engine rankings. *Princeton University Press*.

[12] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 556–559, 2003.

[13] Andrew W. Moore Purnamrita Sarkar. A tractable approach to finding closest truncated-commute-time neighbors in large graphs. In *The 23rd Conference on Uncertainty in Artificial Intelligence(UAI)*, 2007.

[14] Lee Rainie and John Horrigan. Internet: The mainstreaming of online life, pew internet and american life project. 10 Feb. 2005, p. 64.

[15] Cartic Ramakrishnan, William H. Milnor, Matthew Perry, and Amit P. Sheth. Discovering informative connection subgraphs in multi-relational graphs. *SIGKDD Explor. Newsl.*, 7(2):56–63, 2005.

[16] Paul Resnick and Hal R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, 1997.

[17] Philip H Sherrod. *DTREG: Predictive Modeling Software*. Software available at http://www.dtreg.com.

[18] Isabelle Stanton, Nina Mishra, Robert Schreiber, and Robert Tarjan. Recommending groups in social networks. Unpublished Manuscript, 2008.

[19] Hanghang Tong, Christos Faloutsos, and Yehuda Koren. Fast direction-aware proximity for graph mining. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 747–756, 2007.

[20] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions, 2003.