Detecting Conversing Groups of Chatters: A Model, Algorithms, and Tests

Seyit Ahmet Çamtepe
camtes@cs.rpi.edu

Mark Goldberg
goldberg@cs.rpi.edu

Malik Magdon-Ismail
magdon@cs.rpi.edu

Mukkai Krishnamoorty
moorthy@cs.rpi.edu

Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180, USA.
January 21, 2005

Abstract

Chatrooms, for example Internet Relay Chat, are generally multi-user, multi-channel and multi-server chat-systems which run over the Internet and provide a protocol for real-time text-based conferencing between users all over the world. While a well-trained human observer is able to understand who is chatting with whom, there are no efficient and accurate automated tools to determine the groups of users conversing with each other. A precursor to analysing evolving cyber-social phenomena is to first determine what the conversations are and which groups of chatters are involved in each conversation. We consider this problem in this paper. We propose an algorithm to discover all groups of users that are engaged in conversation. Our algorithms are based on a statistical model of a chatroom that is founded on our experience with real chatrooms. Our approach does not require any semantic analysis of the conversations, rather it is based purely on the statistical information contained in the sequence of posts. We improve the accuracy by applying some graph algorithms to clean the statistical information. We present some experimental results which indicate that one can automatically determine the conversing groups in a chatroom, purely on the basis of statistical analysis.

1 Introduction

Internet chatrooms are means for information exchange among millions of users around the world. For example, there are at least 785 IRC (Internet Relay Chat) networks distributed all around the world. There are total of 5798 servers within these networks having 1,344,741 users on 679,083 channels [4, 6, 7, 8, 9, 5]. A channel (or equivalently a chatroom) is a community of users who receive all messages addressed to the channel, i.e., messages posted by any user can be viewed by all the other users on the channel. On account of the anonymity offered by such open forums, a host of undesirable activity has evolved, for example kidnappers luring young victims by posing as fake identities and malicious groups covertly planning undesirable acts. In order to battle the onslaught of undesirable behavior, the need for automated tools to study internet chatrooms is clearly needed. A first step is to determine who is chatting with whom. A glance at the sample chatroom log shown below indicates that even to the well trained human eye, this is not a trivial problem. Further any form of automated semantic analysis is tedious and likely to be unsuccesful on account of the extremely unstructured lexicon used. We thus focus on a statistical approach.

[20 : 01 : 18] <id_1> I shall powerful fart from apple pie, than from hamburger
[20 : 01 : 41] <id_2> some girls who want to chat with a male with webcam??
[20 : 01 : 55] <id_3> I shall powerful fart from mom beans than from american taco
[20 : 01 : 57] <id_3> hahahahahahahahah
[20 : 02 : 18] <id_4> hey <BB> id_{11} <AB> : i'm happy :P
[20 : 03 : 35] <id_1> Big farting from salad Olivier!
[20 : 03 : 40] <id_5-> be nice id_4
[20 : 03 : 47] <id_5-> or be gone
[20 : 04 : 18] <id_7> still searching for guys between 35 nd 45
[20 : 04 : 23] <id_4> <BB> id_5- <AB> : did i talked to yu
[20 : 04 : 26] <id_4> did i say somthing bad
[20 : 04 : 30] <id_8> id_9@hotmail.com
[20 : 04 : 30] <id_8> id_9@hotmail.com

The essential features of a IRC chatrooms are that a user in a particular chatroom posts to that particular chatroom by sending a message to a server for the chatroom. The message is then viewable by all users in that particular chatroom. We refer to the references for a detailed discussion of IRC chatrooms. Each post has three tags associated with it, <t,id,message>: t is the time of the post; id is the ID posting the message; and message is the message that was posted. The question we address is whether it is possible to identify what the conversing groups are based only on the times and IDs of the posts, *i.e.*, ignoring the actual message texts. We denote a conversation to be a group of chatters that are all chatting on a single topic.

We propose a model of a chatroom, and efficient algorithms to discover all the conversations that are occuring in the chatroom. Our model is based on the following generally observed dynamics in a typical chatroom: When a user joins a channel, s/he (i) selects one of the existing subtopics and joins into that discussion, or (ii) generates his/her own subtopic and tries to coerce people to talk about that subtopic by inviting them, challenging them, etc. In principle, a user can be in more than one subtopic, but generally s/he will only be active in one of the subtopics. Thus, it is reasonable to assume that a user can only be in one subtopic at any time. A user in a subtopic

might get bored or may simply get charmed by another subtopic and therefore switch subtopics. However, for a short enough time period, a user will stay in one subtopic.

Our experiments indicate that modulo our model assumptions, our algorithms accurately extract the conversations, with both low false negatives and low false positives. Our algorithms extend the ideas first developed in [2] for the problem of identifying users in an open forum that post under different multiple ID's.

Our Contribution. We present a model for chatrooms and novel, efficient algorithms for determining conversations. Our model does not use semantic information in any way. We design two algorithms which we rigorously test using simulation on our model. Both algorithms treat the chatters as nodes in a graph. A hyperedge is a collection of chatters in a conversation. Our assumption that chatters belong to only one subtopic implies that the hyperedges are disjoint. The first step in both algorithms is to use statistical information on the posts to create a candidate hyperedges. We then "clean" the hyperedges obtained from the statistical algorithms using transitivity (which translates to connectivity in the graph) and graph coloring.

Related Work. There are several preliminary works on analysing chatroom conversations. PieSpy [11, 12] is an IRC program that connects to IRC servers and listens to IRC channels. It collects messages and extracts information to visualize social networks in a chatroom. It has simple set of heuristic rules to decide who is talking to who. These rules include direct addressing (destination nickname is written at the beginning of the message). Direct addressing is simple and usually a reliable method, however, it is rarely the case that the destination nickname is used in a post. Temporal Proximity is another approach. If there is a long period of silence before a user sends a message and this message is immediately followed up by a message from another user, then it is reasonable to imply that the second message was in response to the first. Temporal Density is an approach when Temporal Proximity is not applicable. Basically, If several messages have been sent within a time period all from only two users, then it is assumed that there is a conversation between these two users. However, this idea breaks down when there are many conversations simultaneously occuring.

Chat Circle [13] is an abstract graphical interface for synchronous conversation which aims to create a richer environment for online discussions. This approach is based on graphical visualization and does not provide eavesdropping capability. Social Network Analysis (SNA) [10] is a software tool which considers relationships between people, groups, organizations, computers or other information/ knowledge processing entities. The nodes in the network are the people and groups while the links are the relationships between the nodes. SNA is based on computing the metrics: degrees, betweenness, closeness, boundary spanners, peripheral players, network centralization, structural equivalence and cluster analysis. Since defining an accurate graph for IRC is a hard problem, the results based on a graph construction may have high noise.

Camtepe *et all.* [1] present an automated surveillance system for data collection and analysis in Internet chatrooms based on SVD techniques.

The rest of the paper is organized as follows: in section 2 we describe the model, in section 3 we give the algorithms. In section 4 we give detailed results and conclude in section 5.

TNT	**T**otal **N**umber of sub**T**opics
TNU	**T**otal **N**umber of **U**sers
SST	**S**imulation **S**tart **T**ime
SET	**S**imulation **E**nd **T**ime

Table 1: Model: Parameters

2 The Model

In this work we model a single chatroom. Chatrooms usually have a general topic and members of the chatrooms form small groups and talk on one or more subtopics. Messages sent by all users in all subtopics are seen by all other users. In this model we assume that:

1. there is a fixed number of subtopics which are created at the beginning and never halt throughout the simulation;
2. the total number of users in the channel during the simulation remains the same;
3. users have certain life time within the channel, *i.e.* each user has a start time and end time which are selected uniformly at random, they appear in the channel during that single time period;
4. when a user arrives to channel, he/she uniformly at random selects a subtopic and stays in that subtopic during his/her lifetime in the channel,
5. at any time only one user may be selected for posting message in a subtopic;
6. user who is selected for posting has to wait for *message interarrival time* before *actually sending* the message, next user to send will be selected at the time of this actual post;
7. *message sizes* and *message interarrival times* are random at a given distribution and mean; and
8. messages posted from each subtopic at any time are merged and shuffled uniformly at random.

Based on these assumptions, our model has parameters given in Table 1 and variables given in Table 2.

Each user's start time (STU) and end time (ETU) are decided in one of the three ways: (i) every user joins when simulation starts an leaves when simulation ends, (ii) users' join and leave times are uniformly distributed over the interval SST to SET, (iii) simulation time domain is divided into two or more regions, every user joins at a time which is uniformly at random selected in the first region and leaves at a time which is uniformly at random selected in the last region.

Note that at any time t_t, a user might be entering or leaving a subtopic. Therefore subtopic populations will be changing with time. Message posts within the subtopic will be regulated by

t_t	t^{th} time unit
U_i	**U**ser **i**
STU_i	**S**tart **T**ime for U_i
ETU_i	**E**nd **T**ime for U_i
T_j	sub**T**opic **j**
$MT_{j,t}$	**M**ember set of T_j at t_t
$N_{j,t}$	**N**umber of users in $MT_{j,t}$, i.e. $N_{j,t} = \|MT_{j,t}\|$
PHL_j	**P**osting **H**istory **L**ist of subtopic **j**
$PM_{i,j}$	**P**robability **M**ultiplier for user **i** in subtopic **j**
$PP_{i,j}$	**P**robability of **P**ost for user **i** in subtopic **j**

Table 2: Model: Variables

using $k - step$ probabilities. Basically, each user i in a subtopic j at time t_t will have a probability of post $PP_{i,j}$ where $1 \leq j \leq TNT$, $1 \leq i \leq |N_{j,t}|$ and $N_{j,t} = |MT_{j,t}|$, and where:

$$\forall T_j (1 \leq j \leq TNT), \sum_{\forall i: U_i \in MT_{j,t}} PP_{i,j} = 1$$

In *k-step* probabilities, there are $k + 1$ probabilities for posting. We hold a Posting History List (PHL) of size k. Basically, the one sending last is pushed into front. If the size of the PHL exceeds k then, the one at the end is removed. It is possible that a user within the PHL has quit the subtopic, in that case, the user must be marked and should not be assigned a probability. All other users must preserve their places in the list.

To assign posting probabilities; each user is assigned a probability multiplier. The one at the front takes multiplier of $c \times 1$ for a constant c, the second one at the front takes multiplier of $c \times 2$ and so on. All users who are not in the list are assigned same multiplier of $c \times (k + 1)$. To obtain probabilities, the multipliers are divided by the sum of all multipliers. This guarantees that probabilities adds up to 1.

We employ an *Event Driven Simulation* approach. Table-3 lists events and operations triggered by these events. Each topic may create zero or more of the events 1 and 2, but can create at most one event 3 at a given time unit. If events 1,2 and 3 of a topic occurs at the same unit: execution order of operations triggered by these events are 1, 2 and 3. In this case, new coming users will be considered in selection of the next user to send. *Select next user to post* operation includes: (i) selection of a user based on their probability of posts, (ii) uniformly at random selection of an interarrival time , (iii) generating message, (iv) creating *Message post* event for interarrival time unit later but before or at ETU_i, (v) updating PHL for the id of the user who posted lately and (vi) updating posting probabilities.

Table-4 includes a small part of the message log generated by our model. Each message includes only two information: (i) time of the post and (ii) sender of the message. Our algorithms works on these logs to generate groups talking on subtopics within a chatroom. Our model also generates user to subtopic mapping as given in Table-4. We use these mappings to check results of our

Event ID	Event	Operations Triggered
1	User Join	Select a topic Update member set of the topic (MT) If first user, select next user to post If not first user, update post probabilities
2	User Leave	Update member set of the topic (MT) Update post probabilities
3	Message Post	Output the message Select next user to post if previous one is itself

Table 3: Events and Operations

No	Message (Time and Sender)		Subtopic	Members
1	TIME 6 USER 20		1	15
2	TIME 7 USER 15		2	20, 41
3	TIME 9 USER 61		3	61
4	TIME 12 USER 41		4	24
5	TIME 12 USER 24		...	...

Table 4: Sample Messages and User-subtopic Membership

algorithms. For example users *20* and *41* are both talking on the same subtopic and we expect that our algorithms will figure this out.

3 Algorithms

For our experiments we designed two algorithms that discover communication groups (*subtopics*) in the Internet chatrooms: *Connect* and *Color_and_Merge*. Both algorithms use a subroutine called *Cluster*. The input to *Cluster* is a sequence of posts; its output is two sets of pairs of chatters, called Red and Blue. The pairs in Red (resp. Blue) are viewed as colored red (resp. blue). Every pair of chatters is colored either blue or red.

Cluster declares all red (resp. blue) pairs of chatters as not engaged (resp. engaged) in a conversation. Such an output is incomplete since it does not identify subsets of users that converse, *subtopics*. Furthermore, the information given by *Cluster* may be contradictory. If a, b, and c are chatters from the input, and pairs (a, b) and (b, c) are blue, but pair (a, c) is red, then at least one of these three pairs is incorrectly identified. The goal of the procedures *Connect* and *Color_and_Merge* is to reconcile possible contradictions and produce a complete output. They do it based on different assumptions. Both algorithms use as the input the coloring of the pairs produced by *Cluster*.

Connect views set Blue as the edge-set of a graph B defined on the set of all chatters. The procedure uses the breadth-first search to construct all connected components of B. These components are defined to be the topics of the chatroom. Thus, the idea of *Connect* is to "trust" blue edges: if pairs (a, b) and (b, c) were determined by *Cluster* as blue, then *Connect* "decides" that a, b

and c belong to the same topic, even if pair (a, c) is red.

Procedure *Color_and_Merge* "trusts" red edges more than blue ones. It consider the set of red pairs as the edge-set of another auxiliary graph R, and applies a vertex coloring procedure to split the set of vertices into a number of subsets, color classes, so that no two vertices in a class are adjacent. The vertex coloring problem is a classical NP-hard optimization problem (see, [3]); we use a well-known heuristic *Greedy* to find an approximate solution. The output of *Greedy* is then modified by procedure *Merge*, which replaces some pairs of classes with their union. For this purpose, *Merge* repeatedly examines pairs of current color classes in some order. For each such pair, C_1 and C_2, the ratio $tr = e_b/(|C_1|.|C_2|)$ is computed, where e_b is the number of blue edges (x, y) with $x \in C_1$ and $y \in C_2$. Classes C_1 and C_2 are merged in one class if $tr \geq 0.7$. Thus for $tr \geq 0.7$ all red edges connecting C_1 with C_2 are ignored. Notice, that the threshold value 0.7 was found in learning simulations. The resulting color classes are then declared to be the topics of the chatroom. Below we describe *Cluster*.

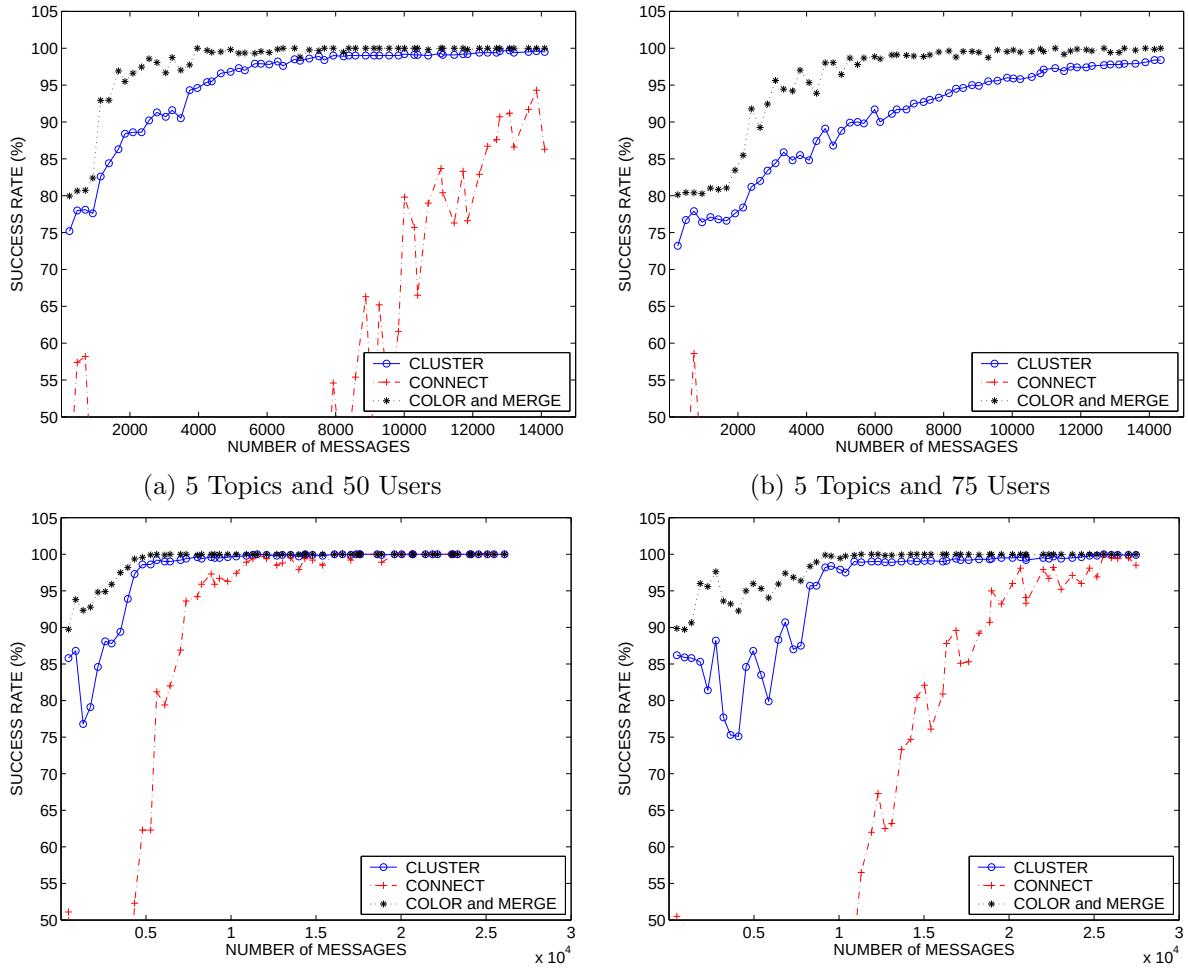
The input to *Cluster* is a message-log of a chatroom. The first stage of the procedure is to find the *minimal interarrival time* $\omega_{i,j}$ for each pair (i, j) of chatters. Note that, we can find minimum interarrival time for a pair (i, j) if both users i and j coexist in the chatroom during some time interval τ. We claim that $\omega_{i,j}$ of two users talking on the same subtopic can not be smaller than a threshold value. Because it takes time for a user to read, interpret, prepare answer, type the answer, and post it. In addition, network and server latencies should also be added. Messages of two users may appear in the chatroom almost simultaneously if they are talking on different subtopics and if they are not responding to each other. Thus, if two different chatters have "small" minimal interarrival time it is likely they belong to different subtopics. To avoid solving the difficult problem of determining what should be considered *Cluster* splits the set of all $\omega_{i,j}$ into two subsets using the 2-mean clustering algorithm.

Procedure 2-mean.

1. $Red = \emptyset$; $Blue = \emptyset$;
2. $Left = 0; Right = \max_{i,j} \omega_{i,j}$;
3. while (new $Left$ and $Right$ are different from the old ones)
4. { for every i, j
5. if $|\omega_{i,j} - Left| \leq |\omega_{i,j} - Right|$
6. $Red = Red \cup (i, j)$
7. else
8. $Blue = Blue \cup (i, j)$;
9. $Left = mean_{(i,j)\in Red}\omega_{i,j}$; $Right = mean_{(i,j)\in Blue}\omega_{i,j}$;}

4 RESULTS

We have implemented and evaluated two algorithms over the chatroom logs generated by our model. Algorithms are run over the logs of various sizes. As explained in the last section, the subroutine, *Cluster*, simply divides pairs of users into two clusters, those talking in the same subtopic and those talking in different subtopics. When a pair of users, who are chatting in the same topic, are output



(a) 5 Topics and 50 Users

(b) 5 Topics and 75 Users

(c) 10 Topics and 50 Users

(d) 10 Topics and 75 Users

Figure 1: Results of the three algorithms on the model for different topic and user number

correctly as being in the same group, it is considered as a correct result. We have counted such correct results and found average success rate over multiple runs. First and second, *Connect* and *Color_and_Merge* respectively, output the groups of the chatroom. To see how these algorithms can be compared for the chatrooms of different sizes and number of topics, we have conducted our experiments with four different chatrooms: (i) Chatrooms with 5 topics and 50 users, (ii) chatrooms with 5 topics and 75 users, (iii) chatrooms with 10 topics and 50 users, and (iv) chatrooms with 10 topics and 75 users.

Parameters of the model are tuned according to observations and statistical analysis which are made over real chatroom logs. Message interarrival times within a topic are uniformly distributed over the interval 3 to 7 time units. So, average message interarrival time within each topic is 5 time units. Average message interarrival time along with the total simulation time decides the number of messages (or log size). Assuming that there are TNT topics, and each topic generates a message

at every 5 time units, there will be average of $TNT/5$ message posts in every time unit. To generate desired number of messages of M total simulation time must be set to $5M/TNT$ time units. It is critical to have enough number of postings by every pair of users to be able to test algorithms effectively. For any pair of users, there must be enough overlap time during which both users exist in the chatroom. Thus, the simulation time is divide into three equivalent intervals where all users arrive in the first interval and leave in the last.

Figure-1 summarizes the experimental results for four different topic and user combinations. First of all, in all experiments, *Color_and_Merge* algorithm provides the best results where the success rate converges to 100% very quickly. When we keep number of the topics constant and increase the number of users (for all algorithms) we see that larger logs are required to provide the same success rate. This is the natural result of increasing the number of users, therefore decreasing the number of messages posted by each user. For example, when the number of users is 50, *Color_and_Merge* algorithm reaches to success rate 100% with a message log of size 4000. But, when there are 75 users, message log of size of 6000 is required to provide same success rate. Other algorithms show the same trend but as seen in Figure-1(b) *Connect* is the most sensitive algorithm to number of messages. In *Connect*, a single false edge is enough to connect two components yielding large number of false results. Therefore, as the number of messages posted by each user decreases, *Connect* fails faster to produce correct results. Finally, if we consider the first 15000 messages, we can see that an increase in the number of topics do not affect the performance of the algorithms.

5 Discussion

We presented a model for a chatroom for which we have showed that it is possible to accurately determine the conversations. While our algorithms are taylored to some extent for the specific model we have presented, the ideas can be generalized to more elaborate models. The advantage of developing models is that algorithms can be rigorously tested on the models. If the model accurately represents the key features of live chatrooms, then one can expect that the algorithms will perform well on the real data.

Ongoing research is to enhance the model, allowing users to belong to multiple conversations (say two), and introducing dynamics by allowing users to switch between conversations. Finally, we would like to apply the algorithms to real data, however one of the main challenges is the scarcity of real data in which the conversations have already been identified – such a task would have to be done by trained human experts.

References

[1] S. A. Camtepe, M. S. Krishnamoorthy, and B. Yener. A tool for internet chatroom surveillance. In *NSF/NIJ Symposium on Intelligence and Security Informatics (ISI 04)*, Tucson, AZ, 2004. ISI.

[2] H.-C. Chen, M. Goldberg, and M. Magdon-Ismail. Identifying multi-users in open forums. In *2nd NSF/NIJ Symposium on Intelligence and Security Informatics (ISI 04)*, Tucson, AZ, 2004. ISI04.

[3] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 2004.

[4] A. Gelhausen. IRC statistics. http://irc.netsplit.de, 1998. (accessed 23 September 2004).

[5] M. St. Johns. RFC 1413 Identification Protocol, 1993.

[6] C. Kalt. RFC 2810 Internet Relay Chat: Architecture, 2000.

[7] C. Kalt. RFC 2811 Internet Relay Chat: Channel management, 2000.

[8] C. Kalt. RFC 2812 Internet Relay Chat: Client protocol, 2000.

[9] C. Kalt. RFC 2813 Internet Relay Chat: Server protocol, 2000.

[10] V. Krebs. An introduction to social network analysis. www.orgnet.com/sna.html, 2004. (accessed 10 September 2004).

[11] P. Mutton. Piespy social network bot. www.jibble.org/piespy/, 2001. (accessed 15 September 2004).

[12] P. Mutton and J. Golbeck. Visualization of semantic metadata and ontologies. In *In: Seventh International Conference on Information Visualization (IV03)*. IEEE, 2003.

[13] F. B. Viegas and J. S. Donath. Chat circles. In *ACM SIGCHI (1999)*. ACM, 1999.