A Tool for Internet Chatroom Surveillance

Ahmet Çamtepe, Mukkai S. Krishnamoorthy, and Bülent Yener *

Department of Computer Science, RPI, Troy, NY 12180, USA.
{camtes, moorthy, yener}@cs.rpi.edu

Abstract. Internet chatrooms are common means of interaction and communications, and they carry valuable information about formal or ad-hoc formation of groups with diverse objectives. This work presents a fully automated surveillance system for data collection and analysis in Internet chatrooms. The system has two components: First, it has an eavesdropping tool which collects statistics on individual (chatter) and chatroom behavior. This data can be used to profile a chatroom and its chatters. Second, it has a computational discovery algorithm based on Singular Value Decomposition (SVD) to locate hidden communities and communication patterns within a chatroom. The eavesdropping tool is used for fine tuning the SVD-based discovery algorithm which can be deployed in real-time and requires no semantic information processing. The evaluation of the system on real data shows that (i) statistical properties of different chatrooms vary significantly, thus profiling is possible, (ii) SVD-based algorithm has up to 70-80% accuracy to discover groups of chatters.

1 Introduction and Background

Internet chatrooms provide for an interactive and public forum of communication for participants with diverse objectives. Two properties of chatrooms make them particularly vulnerable for exploitation by malicious parties. First, the real identity of participants are decoupled from their chatroom nicknames. Second, multiple threads of communication can co-exists concurrently. Although human-monitoring of each chatroom to determine *who-is-chatting-with-whom* is possible, it is very time consuming hence not scalable. Thus, it is very easy to conceal malicious behavior in Internet chatrooms and use them for covert communications (e.g., adversary using a teenager chatroom to plan an unlawful act). In this work, we present a fully automated surveillance system for data collection and analysis in Internet chatrooms. Our system can be deployed in the background of any chatroom as a *silent listener* for eavesdropping. The surveillance is done in the form of statistical profiling for a particular chatter, a group of chatters or for the entire chatroom. Furthermore, the statistical profiles are used to devise algorithms which can process real-time data to determine chatters and their partners. Thus, the proposed system could aid the intelligence community

* This research is supported by NSF ITR Award #0324947.

network	users	channels	servers
QuakeNet	212005	184108	38
Undernet	144138	50720	38
IRCnet	117820	56550	44
GamesNET	45722	44521	22
BRASnet	43927	16523	34

Table 1. Top five IRC networks

to discover hidden communities and communication patterns within a chatroom without human intervention.

IRC (Internet Relay Chat) [1–5] is the original and the most widely used Internet chat medium. Currently there are *675* IRC networks distributed all around the world. There are total of *5290* servers within these networks having *1219906* users on *591257* channels [6]. IRC is a multi-user, multi-channel and multi-server chat system which runs on a Network. It is a protocol for text based conferencing and provides people all over the world to talk to one another in real time. Conversation or chat takes place either in private or on a public channel called as chat room.

IRC follows a client/server model. Server can be running on many machines, in a distributed fashion on a network, where each server has a copy of the global state information of the entire network. IRC client communicates with an IRC server through Internet. Client logs on to a server, picks a unique nickname and selects one or more channels to join. A channel is a group of one or more users who receive all messages addressed to that channel. A channel is characterized by its name, topic, set of rules and current members. Basically, there are two types of channels: *standard channels* and *safe channels*. Standard channels are created implicitly when the first user joins it, and cease to exist when the last user leaves it. Creation and deletion of the safe channels are made by the servers upon request and that is why they are named as *safe channels*. In order for the channel members to keep some control over a channel, some channel members are privileged and called as channel operators. They can perform actions such as setting the topic, ejecting (kicking) a user out of the channel, banning a users or all users coming from a set of hosts, and sharing their privileges with other users.

Server is part of a global IRC server network. When a client sends a message to a channel, server sends this message to all other servers and each server submits the message to people who are part of the same channel. This way, users of a channel, who are distributed all over an IRC network, can communicate in broadcast manner within a channel. In this work, we select Undernet because it is one of the biggest IRC networks with its *38* servers, *50000* channels and more than *100000* users. Table 1 lists top five IRC networks in the world [6].

1.1 Multiple Nicknames Authentication in IRC

IRC is not a secure protocol since all the messages and other information are transmitted in clear text. Authentication is also not reliable since servers authenticate clients: (i) by plain text password, and (ii) by DNS lookups of source IP addresses which are subject to spoof. Multiple nick names further compicate the authenticity of users while analyzing chat room data. During an IRC session a client can change its nickname on the flight. Although such changes are visible from *change nickname* messages, once clients are disconnected, they can use any nicknames they like when they login to server again. Furthermore, a client can access IRC from different hosts under different nicknames. In such situations, it is impossible to decide whether given two nicknames belong to same or different person. We note that some of the channels introduce an additional security by sending an "ident" query to the "identd" service running on client hosts. The Identification Protocol [5] provides a mechanism to determine the identity of a user using a particular TCP connection. Given a TCP source and destination port pair of a TCP socket, Identification Protocol running on client host can first locate the process which has created the socket. Then, it can find the owner of the process and return the user id as the response to the ident query. Identification Protocol itself is not secure and can not be accepted as a trusted authority for this functionality. It is possible to implement a fake *identd* software. Actually, some IRC clients come with their own fake identity services with which client can hide its identity or initiate impersonation attacks.

The multiple nicknames problem is out of the scope of this work. In our system we simply maintain a list of nicknames that a client used in the channel. (Note that this information is available in the messages logs as mentioned above). For a new client, we check nickname against these nickname lists, if it is in a list of a person, we assume that person has disconnected and connected again.

1.2 Related Work

In this work, we focus on IRC networks and build a system for data collection and identification of hidden communication patters and groups. There are several spy tools to listen IRC channels. PieSpy [7, 8] is an IRC program (bot) that logins to IRC servers, joins and listens to IRC Channels. PieSpy is used to collect messages and extract 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 most reliable method to set relation between users, but it can not be used always. *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 Proximity searches for such cases to infer a relationship between the users. *Temporal Density* is an approach when Temporal Proximity is not applicable. Basically, If several messages have been sent within a time period all

$ID \backslash Time$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	1	2	0	1	0	0	0	0	0	1	0	0	2
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	1	2	0	1	0	1	0
7	0	0	0	1	0	1	0	1	0	1	0	2	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	1	0	0	3	1	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2. Example Data Matrix

of them are originating only from two users, then it is assumed that there is a conversation between these two users. This idea is similar to our fixed window approach; however, there is no verifiable performance study of PieSpy nor the impacts of these parameters to the discovery of communication patterns.

Chat Circle [9] 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.

There are several proposals for discovering hidden groups and communication patterns in social networks (see [10–12] and references therein). For example 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.

Organization of the paper This paper is organized as follows. In Section 2 we describe how to access and collect data in IRC. In Section 3 we explain a computational discovery algorithm, based on the Singular Value Decomposition (SVD), for locating groups of chatters. In Section 4 we present the experimental results and finally conclude in Section 5.

2 Data Acquisition

In this work, we have implemented a simple IRC client which logs to a server, selects a unique nickname and joins to a given channel. Program silently listens to the channel (i.e., it is a *silent listener*) and logs all the messages.

Our IRC client continuously collects data. In this work, we used ten days of channel logs for the statistical analysis. For our algorithm, we used two busiest

four-hour periods in a day (i.e., from 08:00 to 12:00, and from 20:00 to 24:00). Each four hour period is divided into eight 30-minute intervals called *sampling windows* (e.g., 08:00 to 08:30, 08:30 to 09:00, etc).We also used other sampling window sizes, i.e. $15, 30, 45, 60, 75, 90$-min. We further divide the sampling windows into *time slots* of sizes 15, 30, 45, 60, 75 or 90 seconds. For each time slot, we count number of messages of each user. We construct data matrices C where each row corresponds to a user and each column to a time slot. Entry at c_{ij} in the matrix is the number of messages of user i in time slot j. An example of data matrix is shown in Table 2 assuming the time slot to be 15 sec.

The first column of each row gives an identity number of the chatter. i, j-th entry of the matrix gives the number of messages that chatter i sent during the j^{th} instance (time slots). In this example, the first chatter (identity 1) is sending two messages at the 4^{th} time slot. Similarly the 4^{th} row chatter (identity 7) is sending one message at the 4^{th} time slot. Our algorithm hypothesizes that possibly chatter 1 and chatter 7 are communicating. This hypothesis gets stronger upon observing the message posting in the time slots 6 and 12. Thus, persistence of co-existence of posting chat messages in the time scale indicates a correlation among chatters. We use Singular Value Decomposition algorithm to quantify such relations as explained later in the paper.

2.1 Generating Data Matrices

We use two methods to generate data matrices. In *fixed window* matrix generation method, the sampling windows do not overlap (e.g., 08:00 to 08:30, 08:30 to 09:00, etc). However, there is a chopping of the conversation at the allotted time interval with the fixed window scheme. Thus, we consider *sliding window* method in which the windows overlap as shown in Figure 1. For example, in sliding window method for window overlap of 15 minutes, we generate matrices for time intervals of 08:00-08:30, 08:15-08:45, 08:30-09:00, etc. This method generates sixteen matrices for 4 hour period for window size of 30 minutes. In the experiments, we investigate the impact of both sliding and fixed window methods, as well as the size of time slots on the quality of the solutions obtained by our algorithm.

2.2 IRC Selection and Noise Filters

We decided to listen three channels in Undernet Network. These are channels *USA*, *Philosophy* and *Political* where the language is English. Each of these channels represents a different communication style and different user profiles. Channel *USA* is a recreation channel which contains large amounts of junk messages. Messages are short, conversations are short-lived, and people tend to jump from one short non-brainy conversation to another. We consider messages as *noise* if they have no value in chatroom analysis. We observed the following properties of such messages: (i) they are short broadcasts, (ii) they have sexual, offensive, annoying or abusive content, (iii) they are irrelevant with the topics in the channel, and (iv) they usually do not get any response. Usually, owners of such messages

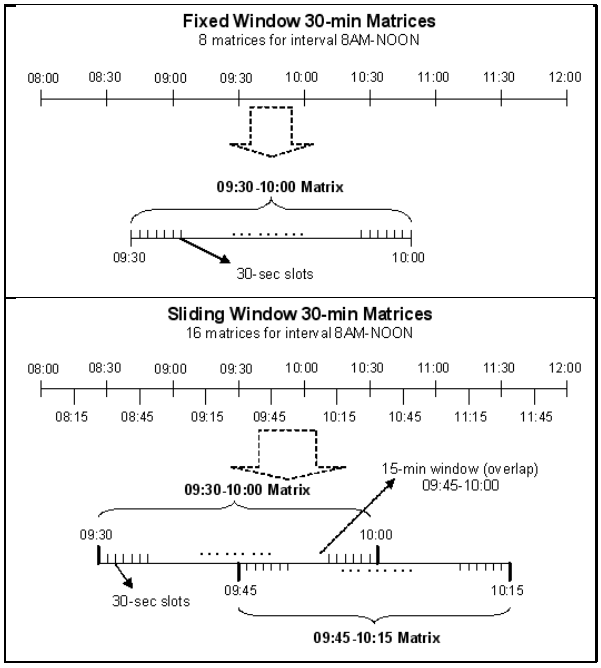

Fig. 1. The Fixed and Sliding Window Time Series

are kicked out of the channel by the channel operators. We apply some simple set of rules to filter out the noise. These rules are based on our preliminary observations and worked fine for our case. First, we eliminate the messages of the *kicked-out-users* with the assumption that they are kicked out because their messages are considered as noise. Next, we eliminate messages of the non-English speakers. This can be achieved by using some frequent non-English (Spanish, Italian, French, German, ...) keywords. We also eliminate users who send messages which include words like *"wanna chat", "cares to chat", "care to chat", "any girls", "some girls", "anyone wants", "anyone who wants", "anyone with webcam", "Chat anyone", "Anyone want to", "someone to talk", "someone want to talk", "someone wanna talk", "someone to chat", "someone want to chat", "someone in usa", "someone from", "looking for someone", "aslpls", "asl pls", "asl please"* Note that such word sequences mean that the chatter is looking for some private chat, and the chatter's intention is not to join a decent conversation within the channel.

Stopword Elimination can be considered as a better technique to clean noise from a chatroom data. Stopwords are a list of common or general terms that appears frequently in a language. Analysis on word frequencies shows that there are few words that appear often and carry really little information. Many search engines don't index stopwords because they're expensive to index, i.e. there

are so many of them, and they carry little useful information. For the case of chatroom analysis, it is possible to generate stopword lists for chatrooms based on the topics, the communication styles, and user profiles.

Channel *USA* has the highest noise among the three IRCs we examined. Channel *Philosophy* is a place where people discuss philosophical subjects with low noise. Communication style here is mostly pairwise, people either select to be part in discussion or to be silent listener and make comments rarely. Finally, in channel *Political*, people discuss political issues. Communication style in this channel is more like groupwise. Two or more users discuss a topic and users may separate into groups according to their political tendency. Discussions occurs among and sometimes within such groups.

3 SVD- based Algorithm for Discovering Chatter Groups

The Singular Value Decomposition has been used to reduce the dimensionality of the data-matrix as well as finding the clusters in a given data-set [13]. In this subsection, we describe a SVD-based algorithm to find the groups in the chat room discussion.

The input to this algorithm is an integer $C_{n\times m} = c_{i,j}$ modeling a chat room conversations where $m \geq n$. The rows are the participants and the columns are the successive time intervals each with length Δ seconds. An entry $c_{i,j} = k$ means that the i-th participant is "talking" at the j-th time interval by posting k messages. Thus the matrix represents conversations for a duration of $m\Delta$ seconds. Since not all the members of a chatroom are chatting all the time the matrix C is quite sparse.

If a set of participants $i_1, i_2, \cdots, i_k$ are *chatting* as a group, we expect them to carry on their conversations over multiple δ units. Thus, there will be many nonzero entries in the corresponding rows for many of the columns. We propose an algorithm to identify such groups.

Algorithm:

Compute SVD

Compute the Singular Value Decomposition of C. This gives three matrices, U, D and V, where $D_{n\times m}$ is a diagonal matrix and $U_{n\times n}$ and $V_{m\times m}$ are orthogonal matrices (i.e., $UU^T = U^TU = I$ and $VV^T = V^TV = I$) such that $C = UDV^T$.

Low Rank Approximation

The diagonal entries of D (by the property of SVD) are nonnegative and decreasing values. By considering the successive rations of the diagonal entries, we chose the first k dimensions. The choosing the first k dimensions contribute an error, which is= $E_k = \sum_{i=1}^{n} d_{j,j}^2 - \sum_{j=1}^{k} d_{j,j}^2$. We normalize the error and choose largest k such that $E_k/E_n \leq \eta$ is minimized for a given relative error bound η.

Clustering the Low Dimension

Cluster the points $d_{1,1}u_1, d_{2,2}u_2, \ddots d_{k,k}u_k$ (in reduced dimension) of the given matrix. Cluster these points (in each of the dimensions) [1].

Report the Clusters

Output the participants in the clusters as groups.

4 Experimental Results

There are several objectives of the experimental study. First, a statistical quantification of chatroom communications is obtained. The statistics are collected both for the entire chatroom and for the pairwise communications as well. Second, we measure the performance of our SVD-based clustering algorithm by checking the accuracy of its results against to a verification tool.

4.1 Statistical Profiling of Chatrooms

We have collected data from three chatrooms over 10-days (10K to 20K messages per day) of time period to determine the message interarrival distribution and message size distributions. In Table 3, we present the overall statistical properties. The table shows that the *USA* chatroom has the most noise and the *Philosophy* chatroom is the least one (based on the second moment information). Similarly the distribution for *USA* chatroom is skewed most (the third moment) and it diverges from the normal distribution the most (the fourth moment information). These two measures indicate long tailed distributions. The table also shows that, in average the messages and the interarrival time of the messages are shortest in the *USA* and longest in *Political*.

In Figure 2, we show the histograms of message interarrival time and message size information for all three chatrooms over 10-days of data. In Figure 3, we apply curve fitting to message size distributions and discover a surprising result that *USA* chatroom message size distribution follows a power law.

Pairwise Chatter Statistics We also collected pairwise information for 4-hour period over the *Philosophy* chatroom by manually studying the data to identify the pairs. The results are shown in Figure 4 for message size, message arrival and conversation duration. A more interesting result is shown in Figure 5 which indicates that conversation duration in *Philosophy* chatroom has a linear decline.

[1] There are several choices of clustering algorithms. In our experiments we applied a thresholding scheme on the first dimension.

	Philosophy	Political	USA	Philosophy	Political	USA
Metric	Msg Size	Msg Size	Msg Size	Arr. Time	Arr. Time	Arr. Time
MEDIAN	49	61	19	61	64	41
AVERAGE	55	68	40	66	69	57
STD DEV	34	38	66	37	34	54
VARIANCE	1210	1466	4456	1430	1216	2965
SKEWNESS	1	1	3	1	1	1
KURTOSIS	1	8	13	2	5	3

Table 3. Chatroom Statistics: Message Size and Interarrival Time

4.2 Verifying and Interpreting the Results

In order to test how good the SVD-based algorithm performs we design a tool which requires a preprocessing of chatroom message logs. In particular, we manually study the 4-hour period over the *Philosophy* and *USA* chatroom data to identify the communicating pairs and to draw a manual-graph. We made several passes over the data by simply reading and interpreting the messages. With the first pass over 2K to 3K messages, we identify communicating pairs, i.e. *who-is-talking-to-whom*. In the subsequent passes, we consider only the identified communicating pairs. For a given pair of user, we implement a simple message filter operation based on the sender nickname and obtained messages belonging to either users. Then, we read those filtered messages and identify messages belonging precisely to the conversation of the given pair of users. Manual-graph consists of edges (communicating pairs) and the nodes (users). Each pairs (edges) has message log of their conversation.

The verification tool takes (i) the edges and nodes of the manual-graph graph, and (ii) clusters of nodes generated by the SVD-based algorithm (i.e. the output of the algorithm). Program first merges clusters sharing at least one node. We assume that these clusters are cliques and we list all possible edges in these cliques. The nodes and edges induced by the clusters of the SVD-based algorithm are compared with the edges and nodes of the manual-graph. Any node which is in the manual-graph but not a member of any cluster produced by the SVD algorithm is assumed to be missed by the algorithm. Similarly, any edge of the manual-graph which is not a member of any cluster is assumed to be missed by the SVD. Program outputs missed and found edges and nodes.

Message logs produced during manual-graph generation process are used to calculate pairwise statistics. They are also used to calculate weights for the edges and nodes of manual-graphs. Weight of an edge is the number of messages exchanged during the conversation of node pairs. Weight of the node is the total number of messages send by this node. Verification program outputs edges and nodes sorted in their weights. Miss of an edge or a node with high weight is not desirable and can be used as a performance metric for evaluating the algorithm.

Time Slots	% of Missed Edges	% of Missed Nodes
15 sec	71 %	69%
45 sec	55 %	44%
90 sec	44 %	28%

Table 4. Philosophy chatroom: 240-min (4-hour) window over 4-Hour data with various time slots

Interpretation We examine the impacts of (i) time slots, and (ii) disjoint vs sliding windows on the performance of the algorithm. In Table 4, we show the results obtained from a large matrix which is obtained from 4-hour data. There is only one matrix corresponding to window of size 240-min (4-hour) over 4-hour data. We experimented different time slots, i.e. 15, 45 and 90 sec. Evidently the longer the time slot the better the results are. Figure 4, presents histogram of message interarrival time for pairwise communications in *philosophy* channel. Interarrival times almost always fall below 90-sec, which we assume is the reason for the performance increase as time slots gets larger. As the time slot gets smaller, number of the columns in the matrix increases yielding more sparse matrix. Moreover, the impact of the noise in the computations is reduced by larger time slots.

In Table 5, we fixed time slot as 30-sec and divided the 4-hour time interval into various window sizes. We would like to examine the impact of time various (15, 30, 90 min) window sizes. Here, we have the same tendency as we reported above. As the window size increases, the performance of the SVD algorithm increases. We claim that results here are strongly related to the duration of the conversation metric presented in Figure 4. Simply, some of the conversations might get chopped at the allotted time interval. Certain duration of conversation values for communicating pairs might create such an increase in the number of chopped messages for certain window sizes. A conversation falling on two or more window might be missed by SVD algorithm. Therefore the larger the window size, the better the results are.

Finally, in Table 6, we compare the performance of our algorithm as a function of different sliding window schemes with fixed time slot of 15-sec. The results show that overlap size does not have any significant impact. The difference among the results of *Philosophy* and *Political* channel is caused by the nature of the chatroom. Since there is a chopping of the conversation at the allotted time interval with the fixed window scheme, our sliding window approach provides an increase in performance of the SVD algorithm. When we compare results in Table 6 and 5, we see that percentage of missed nodes decreases with overlapping; even though time slot used in Table 6 is 15-sec which has been shown to have highest miss rates in Table 4.

Sampling Window	% of Missed Edges	% of Missed Nodes
15 min	47%	36%
30 min	42%	65%
90 min	26%	20%

Table 5. Philosophy chatroom: Non-overlapping windows over 4-Hour data with 30-second time slots

Sliding window overlap	% of Missed Edges	% of Missed Nodes
Phil. 15 min.	42%	28%
Phil. 10 min	42%	28%
Polit. 15 min	58%	41%

Table 6. Philosophy and Political chatroom: 30-Minute overlapping windows (sliding windows) over 4-Hour data with 15 second time slots. The performance of the SVD-based algorithm varies depending on the nature of the chatroom.

5 Conclusions

The Internet chatrooms can be abused by malicious users who can take advantage of its anonymity to plan for covert and unlawful actions. We presented a fully automated system for data collection and discovery of chatter groups and implicit communication patterns in IRC without semantic information or human intervention. We have shown that the chatroom traffic characteristics changes significantly from one chatroom to another. However, quantifying such characteristics enables us to build novel tools that can not only eavesdrop into a chatroom but also it can discover sub-communities and hidden communication patterns within the room.

This work could aid intelligence community to eavesdrop in chatrooms, profile chatters and identify hidden groups of chatters in a cost effective way. Our future work focuses on enhancing the SVD algorithm working in the time domain with topic based information. Our preliminary results show that such enhancement is possible.

References

1. Kalt, C.: RFC 2810 Internet Relay Chat: Architecture (2000)
2. Kalt, C.: RFC 2811 Internet Relay Chat: Channel management (2000)
3. Kalt, C.: RFC 2812 Internet Relay Chat: Client protocol (2000)
4. Kalt, C.: RFC 2813 Internet Relay Chat: Server protocol (2000)
5. Johns, M.S.: RFC 1413 Identification Protocol (1993)
6. Gelhausen, A.: IRC statistics. http://irc.netsplit.de (1998) (accessed 10 February 2004).
7. Mutton, P., Golbeck, J.: Visualization of semantic metadata and ontologies. In: Seventh International Conference on Information Visualization (IV03), IEEE (2003) 300–305
8. Mutton, P.: Piespy social network bot. http://www.jibble.org/piespy/ (2001) (accessed 14 October 2003).
9. Viegas, F.B., Donath, J.S.: Chat circles. In: CHI 1999, ACM SIGCHI (1999) 9–16
10. Krebs, V.: An introduction to social network analysis. www.orgnet.com/sna.html (2004) (accessed 10 February 2004).
11. Magdon-Ismail, M., Goldberg, M., Siebecker, D., Wallace, W.: Locating hidden groups in communication networks using hidden markov models. In: NSF/NIJ Symposium on Intelligence and Security Informatics (ISI 03), Tucson, PA, ISI (2003)
12. Goldberg, M., Horn, P., Magdon-Ismail, M., Riposo, J., Siebecker, D., Wallace, W., Yener, B.: Statistical modeling of social groups on communication networks. In: First conference of the North American Association for Computational Social and Organizational Science (CASOS 03), Pittsburgh PA, CASOS (2003)
13. Golub, G.H., Loan, C.F.V.: Matrix Computations. 3rd edn. The Johns Hopkins University Press, Baltimore, MD (1996)

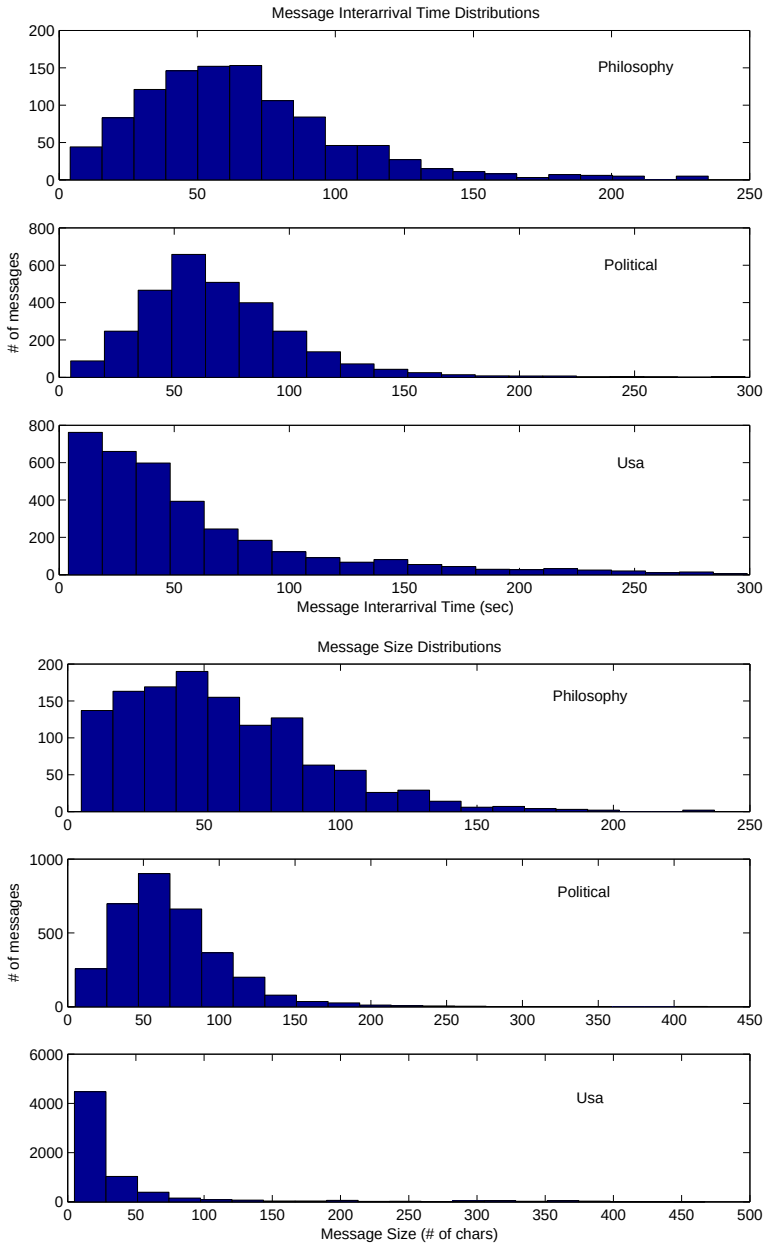

Fig. 2. Message arrival and message size histograms for all three chatrooms

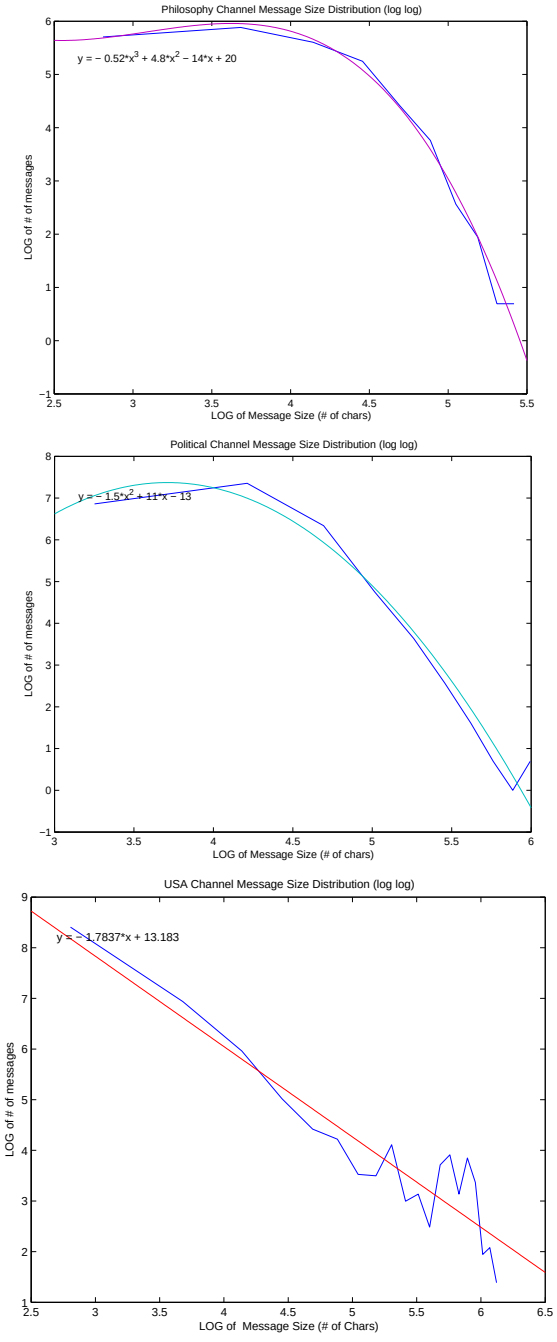

Fig. 3. Message size (log-log) distributions for all three chatrooms. The USA chatroom obeys the power-law.

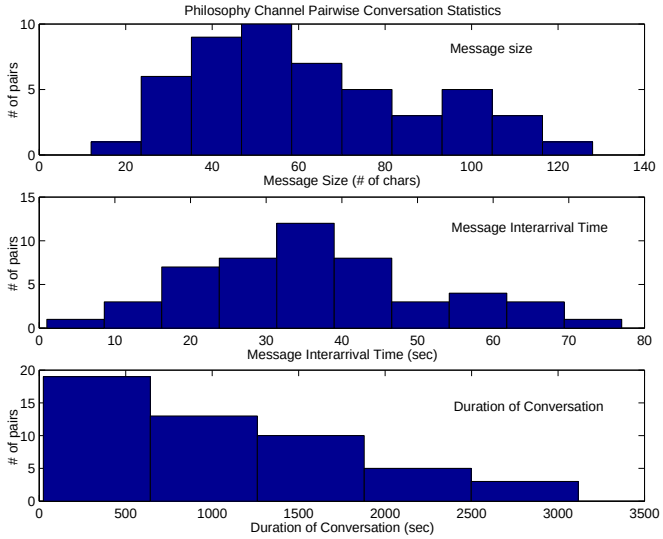

Fig. 4. Pairwise Communication Statistics for *Philosophy chatroom*. The statistics are computed over 50 pairs of chatters during 08:00-12:00 hour period

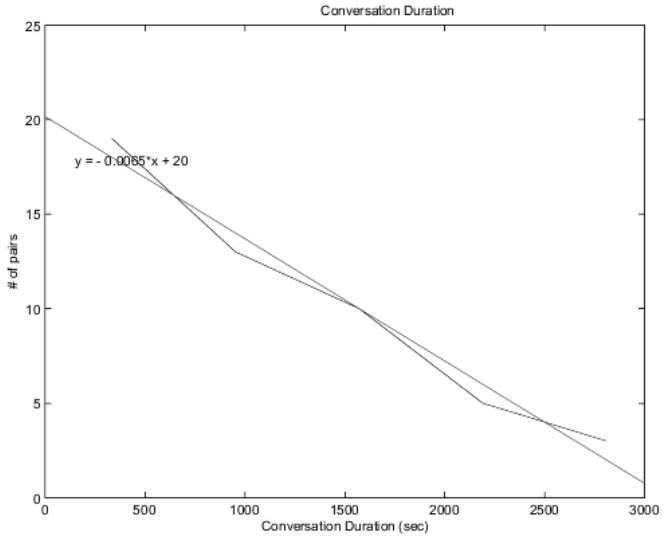

Fig. 5. Pairwise Communication Durations in *Philosophy chatroom* 8:00-12:00.