Lecture Notes in Social Networks

Fazli Can
Tansel Özyer
Faruk Polat *Editors*

State of the Art Applications of Social Network Analysis

Springer

Lecture Notes in Social Networks

For further volumes:
http://www.springer.com/series/8768

Fazli Can · Tansel Özyer · Faruk Polat
Editors

State of the Art Applications of Social Network Analysis

Springer

Editors
Fazli Can
Department of Computer Engineering
Bilkent University
Bilkent
Turkey

Tansel Özyer
Department of Computer Engineering
TOBB University of Economics and Technology
Ankara
Sogutozu
Turkey

Faruk Polat
Department of Computer Engineering
University Campus
Middle East Technical University
Ankara
Turkey

ISSN 2190-5428 ISSN 2190-5436 (electronic)
ISBN 978-3-319-05911-2 ISBN 978-3-319-05912-9 (eBook)
DOI 10.1007/978-3-319-05912-9
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014937954

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

We are proud to present the edited book that contains extended versions of a selected set of papers presented at the *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, *ASONAM 2012*, which was held in Istanbul at Kadir Has University in August 2012. Recently, use of social networks has become a normal activity for a huge number of ordinary people of all ages and as computer scientists we need to find new ways of making their experiences more productive and enjoyable. As a result, social networks and mining have become one of the hottest research fields in computer science and attracts the attention of researchers with different backgrounds. The field is fruitful and offers interesting challenging problems as reflected by our book contents. In the conference there were several top quality papers; however, in this book we are able to include 17 of them written by experts. The studies included in this book consider subjects ranging from practical issues to theoretical considerations such as discovering the political structure in the Italian Parliament, new event detection in tweets, prediction of stock market trends, and finding clique structures in social networks. In the following, we present a short summary of the papers covered in our book. We are confident that the collected efforts presented here will open new horizons for both practitioners and theoreticians working in the field of social networks and mining.

The first chapter: "A Randomized Approach for Structural and Message Based Private Friend Recommendation in Online Social Networks" by Bharath K. Samanthula and Wei Jiang proposes a two-phase private friend recommendation protocol for a particular user by using network structure and message interaction between users. They compute recommendation scores by preserving the privacy of users and for better security they propose an extended version of the proposed protocol using randomization technique. They provide the complexity and security analysis of their protocol and study its applicability based on different parameters.

The second chapter: "Context Based Semantic Relations in Tweets" by Ozer Ozdikis, Pinar Senkul, and Halit Oguztuzun proposes methods to extract semantic relationships among terms in tweets and use them to detect events with higher accuracy, with larger time span, and in a user-friendly form. For discovering semantic associations they use co-occurrence-based statistical methods. They improve their previous work by using similarity scores instead of thresholds and

constant multipliers for semantic expansion and identify context-dependent associations by evaluating terms in specific time windows. Daily event clusters are determined by an outlier analysis.

The third chapter: "Fast Exact and Approximate Computation of Betweenness Centrality in Social Networks" by Miriam Baglioni, Filippo Geraci, Marco Pellegrini, and Ernesto Lastres introduces a novel approach for the betweenness centrality computation. It speeds up considerably Brandes' algorithm. Their approach exploits the natural sparsity of the data to algebraically and efficiently determine the betweenness of those node forming trees (tree-nodes) in the social network. They also give a fast sampling-based algorithm that computes an approximation of the betweenness centrality.

The fourth chapter: "An Agent-Based Modeling Framework for Social Network Simulation" by Enrico Franchi proposes an agent-based modeling framework for simulations over social networks. It is created to deal with large simulations and to work effortlessly with other social network analysis toolkits. In order to allow people without a strong programming background to write simulations easily, the framework comes with an internal Domain-Specific Language (DSL) embedded in Python. Their experience with their approach shows that it is successful in providing a user-friendly environment to perform agent-based simulations over social networks.

The fifth chapter: "Early Stage Conversation Catalysts on Entertainment-Based Web Forums" by James Lanagan, Nikolai Anokhin, and Julien Velcin studies conversation of forum users' posts for television series episodes, and analyzes comments to obtain a description of the principal point of interest, which is referred to as peak. They focus on this peak and evaluate the behavior of users within it compared with during the entire conversation life cycle. They show by their approach that it is possible to identify those users within the forum who act as conversation catalysts for subsequent analysis, hence they do not lose a significant amount of information that has been produced by such members of the community.

The sixth chapter: "Predicting Users Behaviours in Distributed Social Networks Using Community Analysis" by Blaise Ngonmang, Emmanuel Viennet, and Maurice Tchuente studies the churn prediction problem, i.e., predicting the tendency of a user to stop using a social platform. They present a novel algorithm, which can deal with pathological cases, to accurately detect overlapping local communities in social graphs. They show that using the graph attributes it is possible to design efficient methods for churn prediction. They present experiments and a successful implementation of their approach using the Hadoop Hbase framework.

The seventh chapter: "What Should We protect? Defining Differential Privacy for Social Network Analysis" by Christine Task and Chris Clifton studies the use of differential privacy; which is an alternative privacy model, popular in data-mining over tabular data, that uses noise to obscure individuals' contributions to aggregate results, in social networks. They present a practical introduction to the application of differential privacy to social networks, review two existing standards for adapting differential privacy to network data and analyze the feasibility of several common social-network analysis techniques under these standards.

The eighth chapter: "Complex Network Analysis of Research Funding: A Case Study of NSF Grants" by Hakan Kardes, Abdullah Sevincer, Mehmet Hadi Gunes, and Murat Yuksel discovers interesting complex network structures from the NSF funding data, and derive the collaboration among researchers in obtaining federal funding. Collaboration patterns at different periods are extracted at researcher, institution, and state levels.

The ninth chapter: "A Density-Based Approach to Detect Community Evolutionary Events in Online Social Networks" by Muhammad Abulaish and Sajid Yousuf Bhat presents a density-based community mining method for tracking overlapping community evolution in online social networks. It adapts a preliminary community structure toward dynamic changes in social networks using a novel density-based approach for detecting overlapping community structures and automatically detects evolutionary events. Their method does not require the neighborhood threshold parameter to be set by the users; rather it automatically determines it for each node locally. They show that the proposed method is computationally efficient and naturally scales to large social networks.

The tenth chapter: "@Rank: Personalized Centrality Measure for Email Communication Networks" by Paweł Lubarski and Mikołaj Morzy uses a large dataset of email communication within a constrained community to discover the importance of actors in the underlying network as perceived independently by each actor. For this purpose they use the simple notion that people are more likely to quickly respond to emails sent by people whom they perceive as important. They propose several methods for building the social network from the email communication data and introduce various weighting schemes that correspond to different perceptions of importance and compare their results with a ground truth to verify their method.

The eleventh chapter: "Twitter Sentiment Analysis: How to Hedge Your Bets in the Stock Markets" by Tushar Rao and Saket Srivastava studies identifying relationships between Twitter-based sentiment analysis of a particular company/index and its short-term market performance. For this purpose they use more than four million tweets. They show that negative and positive dimensions of public mood carry strong cause–effect relationship with price movements of individual stocks/indices. Their analysis of individual company stocks indicate strong correlation values with Twitter sentiment features of that company. They also investigate other features such as how previous week sentiment features affect the next week's opening.

The twelfth chapter: "The Impact of Measurement Time on Subgroup Detection in Online Communities" by Sam Zeini, Tilman Göhnert, Tobias Hecking, Lothar Krempel, and H. Ulrich Hoppe studies the community detection problem. They consider methods that allow for the detection of overlapping clusters, they are the Clique Percolation Method and Link Community detection. They use these two methods to analyze data like mailing lists from some open source developer communities and compare the results for varied time windows of measurement. They observe that certain minimal window size is needed to get a clear image with enough "light" (i.e., dense enough interaction data) to detect subgroup membership.

The thirteenth chapter: "Spatial and Temporal Evaluation of Network-Based Analysis of Human Mobility" by Michele Coscia, Salvatore Rinzivillo, Fosca Giannotti, and Dino Pedreschi uses complex network techniques to mine and analyze a large dataset of human trajectories of a large GPS dataset of vehicles. They build a multiresolution spatial grid and map the trajectories to several complex networks, by connecting the different areas. They analyze different temporal slices of the network, the structural properties of the temporal and geographical slices, and their human mobility predictive power. They provide pointers regarding the significance of their results in understanding of the data transformation process that is needed to connect mobility with social network analysis and mining.

The fourteenth chapter: "An Ant Based Particle Swarm Optimization Algorithm for Maximum Clique Problem in Social Networks" by Mohammad Soleimani-pouri, Alireza Rezvanian, and Mohammad Reza Meybodi uses the Particle Swarm Optimization algorithm to enhance the performance of the ant colony optimization algorithm for finding the maximum clique in social network graph. It is known that finding a maximum clique is essential for analysis of certain groups and communities in social networks. The simulation results on popular social network datasets and some additional graphs show improved performance with respect to the standard ant optimization algorithm.

The fifteenth chapter: "XEngine: An XML Search Engine for Social Groups" by Kamal Taha introduces a collaborative filtering recommender system called XEngine. It categorizes social groups based on various user characteristics such as age, ethnicity, religion, etc. The output generated by XEngine is ranked according to the user preferences and system inferred user social groups. The experimental comparison with three other systems and statistical tests shows that it provides a marked improvement and in terms of execution time efficiency it is comparable with other systems. A demo version of the engine is made accessible on the Web.

The sixteenth chapter: "Size, Diversity and Components in the Network Around an Entrepreneur: Shaped by Culture and Shaping Embeddedness of Firm Relations" by Maryam Cheraghi and Thomas Schott examines the causal scheme of culture, personal networking, and business networking, where attributes of the entrepreneur and the firm are included as controls. They conceptualize properties of the personal network around the entrepreneur, specify hypotheses about cultural effects on the personal network, test them, and then turn to hypothesizing and testing subsequent effects on the business network around the entrepreneur's firm. Their conclusions indicate that entrepreneurs' personal networks and firms' business networks have consequences for their performance, specifically their innovation and expectations for growth of their firms.

The seventeenth chapter: "Content Mining of Microblogs" by M. Özgür Cingiz and Banu Diri considers the problem of microblog classification. In their work, they use Multinominal Naive Bayes and Support Vector Machines and show that the first one significantly outperforms the second. They also present several other experimental observations and possible uses of their results that may help other researchers in their studies.

Before concluding, we would like to thank our authors for their significant contributions and referees for their insightful constructive criticism. Finally, we would like to acknowledge the invaluable support of several Springer people who made this book possible.

Fazli Can
Tansel Özyer
Faruk Polat

Contents

A Randomized Approach for Structural and Message Based Private Friend Recommendation in Online Social Networks

Bharath K. Samanthula and Wei Jiang

Abstract The emerging growth of online social networks have opened new doors for various business applications such as promoting a new product across its customers. Besides this, friend recommendation is an important tool for recommending potential candidates as friends to users in order to enhance the development of the entire network structure. Existing friend recommendation methods utilize social network structure and/or user profile information. However, these techniques can no longer be applicable if the privacy of users is taken into consideration. In this chapter, we first propose a two-phase private friend recommendation protocol for recommending friends to a given target user based on the network structure as well as utilizing the real message interaction between users. Our protocol computes the recommendation scores of all users who are within a radius of h from the target user in a privacy-preserving manner. We then address some implementation details and point out an inherent security issue in the current online social networks due to the message flow information. To mitigate this issue or to provide better security, we propose an extended version of the proposed protocol using randomization technique. In addition, we show the practical applicability of our approach through empirical analysis based on different parameters.

1 Introduction

Online social networks [4, 22] such as Facebook and Google+ have been emerging as a new communication service for users to stay in touch and share information with family members and friends over the Internet. Since the users are generating huge

B. K. Samanthula (✉) · W. Jiang
Department of Computer Science, Missouri S&T Rolla, Rolla, MO 65409, USA
e-mail: bspq8@mst.edu

W. Jiang
e-mail: wjiang@mst.edu

F. Can et al. (eds.), *State of the Art Applications of Social Network Analysis*, Lecture Notes in Social Networks, DOI: 10.1007/978-3-319-05912-9_1,

amounts of data on social network sites, an interesting question is how to mine this enormous amount of data to retrieve useful information. Along this direction, social network analysis [29, 35, 38] has emerged as an important tool for many business intelligence applications [3] such as identifying potential customers and promoting items based on their interests. In particular, since users are often interested to make new friends, friend recommendation application provides the medium for users to expand his/her social connections and share information of interest with more friends. Besides this, it also helps to enhance the development of the entire network structure.

Discovering new friends to a given target user A is equivalent to solving the link prediction [23] problem for A in the corresponding social network. Given a snapshot of the social network, the link prediction problem aims at inferring the new interactions that are likely to happen among its nodes. In our case, the nodes of the social network are the users and an edge between two users indicates a friendship between them. Briefly, friend recommendations can be performed as follows. (i) Social closeness (hereafter, we refer to it as recommendation score) between A and each potential candidate is computed. (ii) The candidates with Top-K scores are recommended as new friends to A.

In general, recommendation score between any two given users can be computed either based on the network topology and/or user profile contents (such as previous employer, location and hobbies). For the past few years, researchers have been focused on developing hybrid friend recommendation algorithms [5, 20] to take advantages of both approaches. Recently, Dai et al. [9] proposed a new friend recommendation algorithm (denoted as CSM - meaning "Combine Structure and Messages") by utilizing the real messages communicated between the users as well as the network structure. To be concrete, this chapter computes the recommendation scores between users based on the similarity metric given in [9]. More details are given in the later part of this section.

The computation of recommendation scores based on the similarity metric given in [9] is straight-forward if user's data are public. However, as users are more concerned about their privacy [8, 10, 16, 21, 40], many online social networks have provided various privacy settings for users to keep their data private. In general, users are allowed to keep their friend lists, profile information etc., as private information. More specifically, in this chapter, we assume that user's data are encrypted and stored on the server of network provider. We emphasize that this is a commonly made assumption in the related problem domains such as in [2, 7, 14, 26, 36, 37]. We believe that, due to increasing privacy concerns, such a private social network will become more common, and it can attract non-traditional users. Under this scenario, the computation of recommendation scores is non-trivial. Along this direction, we propose a two-phase private friend recommendation algorithm based on the similarity metric proposed in [9]. Our method computes the recommendation scores between A and all potential users who are h-hop away from A in a privacy-preserving manner. Figure 1, shows a sample network for target user *Lee* with $h = 3$. In practice, as proposed by Milgram [28], any two persons can get acquainted each other through six degree of separation (i.e., $1 < h \leq 6$) in the network.

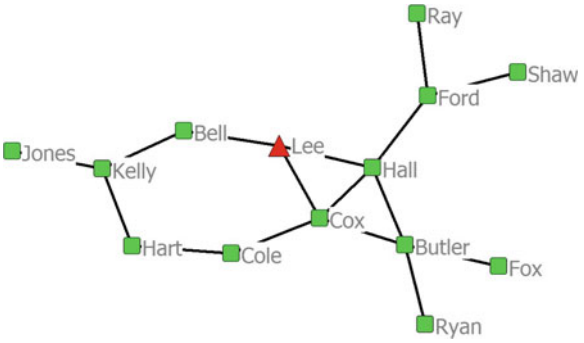


Fig. 1 A sample social network for target user Lee showing users within a radius of 3 (i.e., $h = 3$)

We also present various practical implementation details of the proposed protocol. In addition, we point out an inherent security issue in the current online social networks due to the message flow information between various users. To mitigate this issue or to provide better security, we propose an extension to the proposed protocol using randomization technique.

1.1 Problem Definition

Consider a social network graph G_s with the nodes denoting the users and the (directed) weighted edge between any two nodes denoting the number of real message interactions between them. Since the message interaction can be bi-directional, we take the minimum number of messages, as mentioned in [9, 25], as the actual weight of the edge (denoting the strength of the relationship). A sample minimum message interaction between various users (for $h = 3$) in Lee's network is as shown in Fig. 2. In general, if user A sends n_1 messages to B and B sends n_2 messages to A, then the weight of the edge between A and B is taken as $\min(n_1, n_2)$. This further implies that the weight of the edge between any two friends is directly correlated to the strength of their relationship (i.e., larger weight indicates stronger friendship).

For a target user A (whom we wish to recommend friends), we generate a candidate network with A as the root and an edge between the users denote the number (minimum) of real message interactions. Note that the users who are 1-hop away from A are actually his/her friends. In order to generate the candidate network, we have to remove the links between users at the same level. E.g., refer to Fig. 2, we can generate the candidate network by removing the link between $Hall$ and Cox (since they are on the same level). The recommendation score (RS) between A and any potential user U who is l-hop ($2 \leq l \leq h$) away from A in the candidate network is given as [9]:

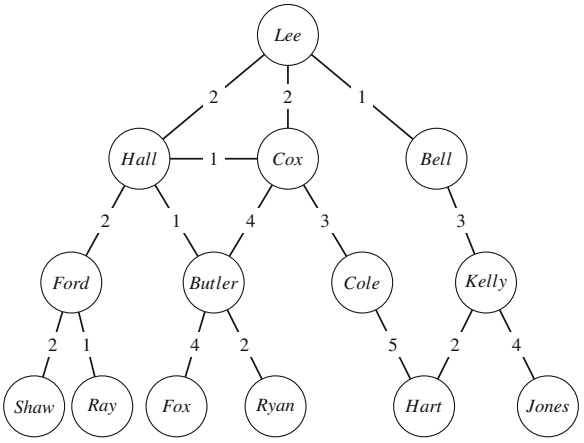


Fig. 2 Message interaction between different users in *Lee*'s network

$$RS(A,U) = \left(\sum_{k}\left(|P_k(A,U)| * \prod_{i} C(S_{i-1},S_i)\right)\right) * \frac{D_U}{TN} \tag{1}$$

where $P_k(A,U)$ denote all the intermediate users on the kth shortest path starting from A (root) to user U, $|P_k(A,U)|$ is the total number of messages along path $P_k(A,U)$, and $L(i)$ is the set of all users at level i (i.e., i-hop away from A). $S_i \in P_k(A,U) \cap L(i)$, for $i = 1,\ldots,l-1$, where $U \in L(l)$. Note that S_0 denote the root user A. $C(S_{i-1},S_i)$ denote the proportion of messages between user S_i and S_{i-1} to the total number of messages at level i. Here, user S_{i-1} is the parent of user S_i in the corresponding candidate network; D_U denote the degree of U and TN denotes the total number of users in the candidate network.

When the privacy of users is taken into consideration, the computation of above mentioned recommendation score is not straight-forward. More specifically, in this chapter, we assume the following private information (PI) for user U:

(1) **PI 1—Friendship**: The friendship between any two users U and V is not revealed to any other user.
(2) **PI 2—Strength of Friendship**: The weight of an edge between U and V, denoted as $C_{U,V}$, is not revealed to users other than U and V.
(3) **PI 3—Degree**: The size of the friend list of U is not revealed to other users.
(4) In addition, the above user's information should not be revealed to the network administrator.

Without loss of generality, let $U_1,\ldots,U_n$ be the set of potential candidates who are at most l-hop ($2 \le l \le h$) away from A. The goal of this chapter is to develop a private friend recommendation protocol which is formally defined as follows:

$$\text{PFR}(A, F(A), U_1,\ldots,U_n) \rightarrow \Gamma \tag{2}$$

where $F(A)$ denote the friend list of user A. Γ is defined as:

$$\Gamma = \{\langle \widetilde{RS}(A, U_1), U_1 \rangle, \ldots, \langle \widetilde{RS}(A, U_n), U_n \rangle\}$$

Here, $\widetilde{RS}(A, U_j)$ is the new recommendation score for U_j which is correlated to the actual score $RS(A, U_j)$ (based on Eq. 1) as below, for $1 \leq j \leq n$:

$$\widetilde{RS}(A, U_j) = M_h * TN * RS(A, U_j)$$

M_h is the normalizing factor for a user at h-hop away from A and TN is the number of users in the candidate network. For any fixed h and A, we observe that M_h and TN are constants (more details are given in Sect. 3). At the end of the PFR protocol, the values of $\widetilde{RS}(A, U_j)$ and U_j, for $1 \leq j \leq n$, are known only to A and the privacy of each user (PI 1, 2, and 3) is preserved. In practice, since the friend lists can be large, the number of scores returned to A can be in hundreds. Therefore, a more effective way is to simply select Top-K users as the final set of friend recommendations.

1.2 Main Contribution

The proposed protocol computes the recommendation scores between a target user A and all potential candidates who are at most l-hop ($2 \leq l \leq h$) away from A in a privacy-preserving manner. More specifically, the main contributions of this chapter are summarized below:

- **Security**—The proposed protocol guarantees that the friend lists, the strength of friendships, and the friend list sizes of each user are kept as private from other users. However, we identify an inherent security issue that may leak valuable information to the network administrator in the current online social networks which is also applicable to the proposed protocol. To mitigate this risk or to provide better security, we also propose an extended version of the proposed protocol using randomization technique.
- **Accuracy**—Our protocols compute the recommendation scores which are scaled by a constant factor $M_h * TN$; therefore, the relative ordering among the scores is preserved. Hence, our protocols guarantee the same kind of effectiveness similar to the CSM method [9]. That is, the final Top-K list of recommended users in our protocols is the same as in [9] and is independent of K.
- **Efficiency**—In our empirical analysis, we show the practical value of PFR through various experiments. Also, we show that the efficiency of the extended version is very close to that of PFR. We observe that the computation costs incurred on the internal users in our protocols are very small; therefore, the proposed protocols are very efficient from internal users perspective.

The rest of the chapter is organized as follows. We discuss the existing related work in Sect. 2. Section 3 presents the new scoring function that preserves the relative

rankings among the recommendation scores. The proposed PFR protocol, along with a detailed security and complexity analysis, is discussed in Sect. 4. We address various implementation details in Sect. 5 and point out an inherent security issue in the current online social networks in Sect. 6. To mitigate this risk or to provide better security, we present an extended version of the proposed protocol in Sect. 7. Section 8 discusses the empirical results based on various parameters. We conclude the chapter along with future work in Sect. 9.

2 Related Work

As mentioned earlier, friend recommendation is a very useful application for both users and the social network provider. Through social recommendations, users are allowed to make new friends; therefore, expanding their social connections. In addition, it helps the social network provider in a way to enhance the development of entire network structure. In general, recommendation scores between any two given users can be computed either based on the network topology [30, 34] and/or user profile contents [39].

Only recently, researchers have focused on developing hybrid friend recommendation algorithms [5, 20] to take advantages of both approaches. As an independent work, Lo and Lin [25] proposed a graph-based friend recommendation algorithm using weighted minimum-message ratio as the scoring metric. This work was later improved in [9] by taking the evolution of entire network into consideration. The computation of recommendation scores based on the metric given in [9] is straightforward when users data are public. However, due to the growing concerns over user privacy [8, 10, 16, 21, 40], many users prefer to keep their profile data (including their friend lists) as private. Along this direction, many online social networks such as Facebook, provide various privacy settings for users to make their data private. Therefore, the above existing methods are not applicable if privacy of users is taken into consideration.

To make friend recommendations possible even in privacy-sensitive environments, this chapter proposes a two-phase PFR protocol based on the similarity metric given in [9]. Furthermore, we address an inherent security issue in the current online social networks. To overcome this issue, we propose an extended version to the proposed PFR protocol.

2.1 Other Existing PFR Protocols

We emphasize that there has not been much work done in developing efficient PFR protocols based on different metrics. Dong et al. [11] proposed a method to securely compute the social proximity between users in a mobile social network. They have used the cosine similarity metric to compute how close two give users are by treating user's location as private.

Table 1 Common notations

HPEnc$^+$	An additive homomorphic probabilistic encryption system
SMP	Secure multiplication
SMPA	Secure multiplication and addition
T	A trusted party (such as network administrator)
$\langle E, D \rangle$	A pair of HPEnc$^+$ based encryption and decryption function
$\langle pk, pr \rangle$	A public and private key pair corresponding to $\langle E, D \rangle$
$F(U)$	Friend list of user U
m	Size of friend list of target user A
$C_{U,V}$	Minimum number of messages exchanged between U and V (or weight of edge between U and V)
$L(i)$	List of all users at level i in the corresponding candidate network
$M_{i-1,i}$	Total number of (minimum) messages exchanged between users at $L(i-1)$ and $L(i)$
$C(S_{i-1}, S_i)$	Ratio of C_{S_{i-1},S_i} to $M_{i-1,i}$
M_l, M'_l	Normalization and scalar factors for a user $\in L(l)$
Φ	List of encrypted scalar factors
L_A, L_{U_j}	Aggregated list of encrypted number of messages computed by A and user U_j

Machanavajjhala et al. [27] analyzed the trade-offs between accuracy and privacy for private friend recommendation algorithms based on differential privacy [12, 13]. Our work is entirely different from theirs since the security guarantee in our chapter is based on the well-known semi-honest security definition of secure multiparty computation (SMC) [17, 41, 42]. In addition, they use a different similarity metric, namely common neighbors [23] whereas our work is based on the scoring metric given in Eq. 1.

3 Order Preserving Scoring Function

The original scoring function [9] given in Eq. 1 contains a rational factor (i.e., $C(S_{i-1}, S_i)$) which varies with i, for $1 \leq i \leq l-1$ and $2 \leq l \leq h$. Therefore, to perform encryption operations, here we define a new scoring function (producing an integer value) based on Eq. 1 such that the relative rankings among the final recommendation scores are preserved. Table 1 presents some common notations used extensively in this chapter.

3.1 Normalization Factor

Given a snapshot of the network for A, we define the normalization factor for a user l-hop (or friend) away from A (where $1 \leq l \leq h$) as:

$$M_l = \prod_{i=1}^{l-1} M_{i-1,i} \tag{3}$$

where $M_{i-1,i}$, denoting the total number of messages exchanged between users at $L(i-1)$ and $L(i)$, is as given below.

$$M_{i-1,i} = \sum_{\substack{U \in L(i-1) \\ V \in L(i)}} C_{U,V}$$

We explicitly assume $M_1 = 1$ since users who are 1-hop from A are already friends of A. For any two potential candidates U and V who are l-hop away from A, we observe that U and V have the same normalization factor.

Example 1 Refer to Fig. 2. Consider the potential candidate $Cole$ who is 2 hops away from Lee. Here, we have $L(0) = \langle Lee \rangle$ and $L(1) = \langle Hall, Cox, Bell \rangle$. The normalization factor for $Cole$ is $M_2 = M_{0,1} = C_{Lee,Hall} + C_{Lee,Cox} + C_{Lee,Bell} = 5$. Note that the normalization factor for $Ford$, $Butler$, and $Kelly$ (who are also 2 hops away from Lee) is the same as Cole. Similarly, we have $M_{1,2} = 13$. By substituting these values in Eq. 3, the normalization factor for users at level 3 is $M_3 = \prod_{i=1}^{2} M_{i-1,i} = M_{0,1} * M_{1,2} = 65$. □

Observation 1. For any user $S_{i-1} \in L(i-1)$ and $S_i \in L(i)$, one can observe that the value of $C(S_{i-1}, S_i)$ is equivalent to $\frac{C_{S_{i-1},S_i}}{M_{i-1,i}}$. Therefore, for a potential user U at level l, the rational factor in Eq. 1 can be simplified as follows:

$$\prod_{i=1}^{l-1} C(S_{i-1}, S_i) = \prod_{i=1}^{l-1} \frac{C_{S_{i-1},S_i}}{M_{i-1,i}} = \frac{1}{M_l} \prod_{i=1}^{l-1} C_{S_{i-1},S_i}$$

3.2 Scalar Factor

Given a target user A and h, we define the scalar factor for a user at level l, for $1 \leq l \leq h$, as follows:

$$M'_l = \frac{M_h}{M_l} = \frac{M_{0,1} * \ldots * M_{h-2,h-1}}{M_{0,1} * \ldots * M_{l-2,l-1}} \tag{4}$$

where M_l is the normalization factor for a user belonging to $L(l)$. In addition, we observe that M'_l is the same for all users who are at same level l. Furthermore, when $l = h$, we have $M'_h = 1$. Similarly, we have $M'_1 = M_h$. From Fig. 2, the scalar factor for $Cole$ is $M'_2 = \frac{M_3}{M_2} = M_{1,2} = 13$.

Claim For any given target user A and potential candidate U who is l hops away from A, we define the new scoring function (denoted as $\widetilde{RS}(A, U)$) as follows:

$$\widetilde{RS}(A, U) = M'_l * \left(\sum_k \left(|P_k(A, U)| * \prod_i C_{S_{i-1}, S_i} \right) \right) * D_U \tag{5}$$

Note that C_{S_{i-1}, S_i} is the weight of edge between parent user S_{i-1} and S_i on the kth shortest path from A to U, for $1 \leq i \leq l-1$. We claim that the scoring function given in Eq. 5 preserves the relative ordering among the recommendation scores of potential candidates.

Proof Based on Eqs. 3 and 4, and by using Observation 1, we can re-write Eq. 5 as below.

$$\begin{aligned}
\widetilde{RS}(A, U) &= \frac{M_h}{M_l} * \left(\sum_k \left(|P_k(A, U)| * \prod_i C_{S_{i-1}, S_i} \right) \right) * D_U \\
&= M_h * \left(\sum_k \left(|P_k(A, U)| * \prod_i \frac{C_{S_{i-1}, S_i}}{M_{i-1,i}} \right) \right) * D_U \\
&= M_h * TN * \left(\sum_k \left(|P_k(A, U)| * \prod_i C(S_{i-1}, S_i) \right) \right) * \frac{D_U}{TN} \\
&= M_h * TN * RS(A, U)
\end{aligned}$$

The values of M_h and TN are constants for any given snapshot of the social network (for a fixed h). Therefore, we emphasize that all the original recommendation scores are multiplied with the same constant $M_h * TN$. This makes sure that the relative orderings among the recommendation scores of the potential candidates based on Eq. 5 are preserved. More specifically, for any two potential users U and V if $RS(A, U) > RS(A, V)$, then the new scoring function guarantees that $\widetilde{RS}(A, U) > \widetilde{RS}(A, V)$ for any fixed h and A.

Example 2 Refer to Fig. 2 and let us consider the case of computing the recommendation score between Lee and Fox. Here, Fox has two shortest paths from Lee; $P_1(Lee, Fox) = \{Lee, Hall, Butler, Fox\}$ and $P_2(Lee, Fox) = \{Lee, Cox, Butler, Fox\}$. The total (minimum) number of messages along the first path i.e., $|P_1(Lee, Fox)|$ is 7. Similarly, $|P_2(Lee, Fox)| = 10$. Along $P_1(Lee, Fox)$, we have two internal users $Hall$ and $Butler$ who are respectively 1 and 2 hops away from Lee. In addition, we have $C_{Lee,Hall} = 2$ and $C_{Hall,Butler} = 1$. Similarly, for the path $P_2(Lee, Fox)$, we have $C_{Lee,Cox} = 2$ and $C_{Cox,Butler} = 4$. Since Fox is 3 hops away from Lee, her scaling factor M'_3 is 1. By substituting the above values in Eq. 5, the recommendation score for Fox is given as:

$$\widetilde{RS}(Lee, Fox) = 1 * [7 * 2 * 1 + 10 * 2 * 4] * D_{Fox} = 94 * D_{Fox}$$

Whereas, the actual recommendation score for Fox, following from Eq. 1, is given by:

$$RS(Lee, Fox) = \left[7 * \frac{2}{5} * \frac{1}{13} + 10 * \frac{2}{5} * \frac{4}{13}\right] * \frac{D_{Fox}}{TN}$$
$$= \frac{1}{65} * 94 * \frac{D_{Fox}}{TN}$$

where D_{Fox} is the degree (size of friend list) of Fox and TN denotes the size of the candidate network. It is clear that $\widetilde{RS}(Lee, Fox) = M_h * TN * RS(Lee, Fox)$, where $M_h = 65$. □

4 The Proposed Protocol

In this section, we present our private friend recommendation (termed as PFR) protocol which computes the recommendation scores between the target user A and all potential candidates who are at most h-hop (> 1) away from A based on Eq. 5. We explicitly make the following assumptions:

1. If $U \in F(V)$, then $V \in F(U)$, and $C_{U,V}$ is known only to U and V. We assume $F(A) = \langle B_1, \ldots, B_m \rangle$.
2. Each user has a unique user ID (for example, Facebook user ID is generally at most 128-bit integer).
3. There exists a third party T (e.g., network administrator) who generates a pair of encryption and decryption function $\langle E, D \rangle$ for A based on the additive homomorphic probabilistic encryption scheme (HPEnc$^+$) such as the Paillier cryptosystem [32]. The corresponding private key pr is known only to T and the public key pk is public. In addition, let N be the group size (usually of 1,024 bits). For any two given plaintexts $m_1, m_2 \in \mathbb{Z}_N$, the HPEnc$^+$ system exhibits the following properties:
   a. **Homomorphic Addition:** $E_{pk}(m_1 + m_2) \leftarrow E_{pk}(m_1) * E_{pk}(m_2) \bmod N^2$;
   b. **Homomorphic Multiplication:** $E_{pk}(m_1 * m_2) \leftarrow E(m_2)^{m_1} \bmod N^2$;
   c. **Semantic Security:** The encryption scheme is semantically secure as defined in [18, 19]. Briefly, given a set of ciphertexts, an adversary cannot deduce any additional information about the plaintext.

We emphasize that homomorphic encryption is not a new topic and has been in the literature for more than 30 years. Though the first homomorphic encryption scheme by Rivest et al. [33] proved to be insecure, researchers have been successful in proposing alternative additive as well as multiplicative homomorphic encryption schemes. We refer the reader to [15] for a detailed survey on homomorphic encryption schemes and their wide-spread use in secure applications. In the cryptographic community, Paillier cryptosystem [32] is a well-known example of HPEnc$^+$ system

mainly due to its probabilistic nature and semantic security guarantees. Over the past several years, Paillier scheme has been used in a wide range of applications such as privacy-preserving data mining [1, 24] and electronic voting [6]. Therefore, in this chapter, we use Paillier cryptosystem as the underlying encryption scheme and develop a novel PFR protocol.

In order to generate the candidate network, we need to omit the messages between users who are at the same level. For example, in Fig. 2, we should not consider $C_{Hall,Cox}$ for computing the recommendation scores in the PFR protocol (as mentioned in [9, 25]). Thus, to explicitly generate the candidate network, we include an initialization step as follows. Initially, A generates a counter $t = h - 1$ and passes it over to his/her friends. Upon receiving the counter, each intermediate user U stores the value of received counter (locally) and also stores the parent user who sent the counter to U (denoted as $Pr(U)$). After this, U decrements the counter by 1 and sends it to his/her friends. This process continues until users at h-hop from A receive a counter of $t = 0$. Since a user can receive multiple counter values, we have the following observation.

Observation 2. Consider user U, who is l-hop away from A and $1 \leq l \leq h$, receiving multiple t values. We address the following two cases:

Case 1: If the counter values are the same, then U has multiple shortest paths (with parents of U on the same level). In this case, U considers one of the parents (can be chosen randomly) as actual parent $Pr(U)$ and any further communication happens only with that parent. E.g., refer to Fig. 2, "$Hart$" receives $t = 0$ from both $Cole$ and $Kelly$. Therefore, he can pick one of them, say $Kelly$, as $Pr(U)$.

Case 2: If U receives different values of t which happens when U receives counters from parents who are at different levels. In this case, U selects one of the parent user who sent the maximum t value as $Pr(U)$. In the PFR protocol, the child users of U (denoted as $Ch(U)$) are users belonging to $F(U) - R(U)$, where $R(U)$ denotes the set of users who have sent a counter value to U. The important observation here is U omits the messages exchanged with the users who have sent smaller counter values (also dumps the corresponding counter). This further implies that, U considers only messages exchanged between him/her and either $Pr(U)$ or $Ch(U)$ (therefore forming a candidate network by omitting messages with users on the same level). An example to this case is user "Cox" (refer to Fig. 2). Here, Cox receives $t = 2$ and $t = 1$ from Lee and $Hall$ respectively. Therefore, Cox treats Lee as the actual parent user and omits $C_{Cox,Hall}$.

At the end of the initialization step, based on Observation 2, each internal user U who is l-hop away from A, for $1 \leq l \leq h$, has the values of t, pk, $Pr(U)$ and $Ch(U)$. Apart from the above initialization step, the proposed PFR protocol mainly consists of the following two phases:

Phase 1—Secure Computation of Scalar Factors: During Phase 1, A computes the list of encrypted scalar factors (denoted as Φ, where Φ_{l-1} denotes the encrypted scalar factor for level l and $2 \leq l \leq h$) in a privacy-preserving manner. This phase utilizes a secure multiplication protocol (only if $h > 3$) as a building block. At the end, only A knows Φ and nothing is revealed to other users.

Phase 2—Secure Computation of Recommendation Scores: Following from Phase 1, A (with Φ as input), T and other internal users jointly compute the recommendation scores of all potential candidates who are l-hop away from A, for $2 \leq l \leq h$. This phase utilizes a secure multiplication and addition protocol as a building block. The final recommendation scores and the corresponding user IDs are revealed only to A and nothing is revealed to other users.

To start with, A chooses the value of h[1] and executes the initialization step as explained earlier. Then, during Phase 1, A decides whether there is a need to take the help of other users in order to generate Φ. If $h = 2$, A computes Φ locally. Otherwise, for $h > 2$, A computes Φ with the help of internal users. After this, during Phase 2, A sends necessary information to B_i along with his/her user ID and Φ, for $1 \leq i \leq m$. Then, each intermediate user U_j receives the necessary information from $Pr(U_j)$, generates his/her encrypted partial scores (only if U_j is not already a friend of A) and sends the encrypted partial scores to A. In addition, if the value of t (stored during initialization step) of U_j is greater than 0, he/she computes the necessary information (for $t > 0$) and sends it to his/her corresponding child friends. After receiving all the encrypted partial scores, A and T involve in a secure multiplication and addition protocol to compute the recommendation scores for each potential candidate U_j. At the end of this step, only A knows the user IDs of all potential friends along with their recommendation scores (computed based on Eq. 5). The main steps of PFR are shown in Algorithm 1. Now, we discuss the steps involved in each of the two phases in detail.

4.1 Phase 1: Secure Computation of Scalar Factors

If the value of h is 2, then only the child friends of A's friends are considered as the potential candidates. Since the scalar factor for users at $l = 2$ is $M'_2 = 1$, A simply sets $\Phi_1 = E_{pk}(1)$ for security reasons. When $h > 2$, A does not have necessary information to compute the encryption of scalar factors (such as M'_3) since the potential candidates can belong to any $L(l)$, where $2 \leq l \leq h$. Therefore, when $h > 2$, A computes Φ, with the help of internal users who are at most $h - 2$ hops away from A. We observe that potential candidates who are at most $h - 2$ hops away from A are sufficient to generate the encryption of all scalar factors because the partial scores of $M_{h-2,h-1}$ are known to users belonging to $L(h - 2)$. Note that, irrespective of the value of h, $\Phi_{h-1} = E_{pk}(1)$ always hold. Phase 1 involves steps 1 to 16 as shown in Algorithm 1.

In order to compute Φ, for $h > 2$, A simply waits for internal users with $t \geq 2$ to send in the aggregated data. To start with, each internal user U_j (including B_i) performs the following operations based on the value of t:

[1] Note that h should always be greater than 1. Because, when $h = 1$, we have $l = 1$ which implies potential candidates who are 1-hop away from A who are already friends of A.

1. Compute $X_{U_j} = E_{pk}(\sum_{i=1}^{s} C_{U_j,V_i})$, where V_i is the child friend of U_j and $s = |Ch(U_j)|$
2. Create a vector L_{U_j} of size $t-1$; sets $L_{U_j}[t-1]$ to X_{U_j}
3. If $t > 2$, U_j receives L_{V_i} from V_i and updates L_{U_j} by aggregating L_{V_i} component-wise as follows, and sends it to $Pr(U_j)$.

$$L_{U_j}[k] = \prod_{i=1}^{s} L_{V_i}[k] \bmod N^2, \text{ for } 1 \leq k \leq t-2$$

The above process forwards the aggregated data at each internal user in a bottom-up fashion. At the end, A receives L_{B_i} from B_i, for $1 \leq i \leq m$. After this, A generates the final aggregated encrypted list (L_A) and proceeds as follows:

1. $L_A[k] = \prod_{i=1}^{m} L_{B_i}[k] \bmod N^2$, for $1 \leq k \leq |L_{B_i}|$, where L_{B_i} denote the aggregated list received from B_i. The observation is $|L_{B_i}| = h-2$, for $1 \leq i \leq m$.
2. Assign the encrypted scalar factor for level h as $\Phi_{h-1} = E_{pk}(1)$. If $h = 3$, sets $\Phi_1 \leftarrow L_A[1]$. Else, let $L_A = \langle E_{pk}(x_1), \ldots, E_{pk}(x_{h-2})\rangle$. Using secure multiplication (SMP) protocol, as shown in Algorithm 2, A and T jointly compute Φ using L_A as below.

$$\Phi_l \leftarrow E_{pk}\left(\prod_{j=1}^{h-l-1} x_j\right), \text{ for } 1 \leq l \leq h-2$$

The SMP protocol is one of the basic building blocks in the field of secure multi-party computation (SMC) [17]. The basic concept of the SMP protocol is based on the following property which holds for any given $a, b \in \mathbb{Z}_N$:

$$a * b = (a + r_1) * (b + r_2) - a * r_2 - b * r_1 - r_1 * r_2 \tag{6}$$

where all the arithmetic operations are performed under $\mathbb{Z}_N$. Given that A has input $E_{pk}(a)$ and $E_{pk}(b)$, the SMP protocol computes $E_{pk}(a * b)$ as the output (which will be revealed only to A) without disclosing the values of a and b to either A or T. The output of Phase 1 is the list of encrypted scalar factors (in order) for each level. More specifically,

$$\Phi_l = E_{pk}(M'_{l+1}), \text{ for } 1 \leq l \leq h-1$$

where M'_{l+1} is the scalar factor for users at $(l+1)$-hop away from A. If the maximum value of h is 6 (sufficient for most situations), the maximum size of L_A is 4. Therefore, Phase 1 is bounded by 2 instantiations of the SMP Protocol.

Theorem 1 *The output of Phase 1 is the list of encrypted scalar factors (in order) for each level. That is, Φ_l is equivalent to the encryption of scalar factor for users at level $l+1$, where $1 \leq l \leq h-1$. Formally,*

Algorithm 1 PFR

Require: pr is private to T, h is private to A, and pk is public; U_j knows t, $Pr(U_j)$ and $Ch(U_j)$ from initialization step

{**Steps 1 - 7 performed by** U_j **with** $t \geq 2$}

$s \leftarrow |Ch(U_j)|$

$X_{U_j} \leftarrow E_{pk}(\sum_{i=1}^{s} C_{U_j,V_i})$, where $V_i \in Ch(U_j)$

$L_{U_j}[t-1] \leftarrow X_{U_j}$

if $t > 2$ and U_j received L_{V_i} from V_i **then**

  $L_{U_j}[k] \leftarrow \prod_{i=1}^{s} L_{V_i}[k] \bmod N^2$, for $1 \leq k \leq t-2$

end if

send L_{U_j} to $Pr(U_j)$

{**Steps 8 - 16 performed by** A **and** T}

$\Phi_{h-1} = E_{pk}(1)$

if $h \geq 3$ **then**

  $L_A[k] \leftarrow \prod_{i=1}^{m} L_{B_i}[k] \bmod N^2$, for $1 \leq k \leq h-2$

  if $h = 3$ **then**

    $\Phi_1 \leftarrow L_A$

  else

    Compute Φ using L_A as input to the SMP protocol

  end if

end if

{**Steps 17 - 21 performed by** A}

for all $B_i \in Ch(A)$ **do**

  $\alpha_1 \leftarrow E_{pk}(C_{A,B_i})$

  $\alpha_l \leftarrow \Phi_{l-1}^{C_{A,B_i}} \bmod N^2$, for $2 \leq l \leq h$

  send A, Φ, and α to B_i (note that α is different for each B_i)

end for

{**Steps 22 - 36 performed by** U_j}

if $A \in F(U_j)$ **then**

  send A, Φ and α to each $V_i \in Ch(U_j)$

else

  compute $\beta_j \leftarrow \alpha_1^{D_{U_j}} \bmod N^2$

  compute $\gamma_j \leftarrow \alpha_2 * \Phi_1^{C_{Y,U_j}} \bmod N^2$

  $Z_j \leftarrow \{E_{pk}(U_j), \langle \beta_j, \gamma_j \rangle\}$

  send Z_j to A

end if

if $t > 0$ **then**

  $\Phi_l \leftarrow \Phi_{l+1}$, for $1 \leq l \leq t$

  $\alpha_1 \leftarrow \alpha_1^{C_{Y,U_j}} \bmod N^2$

  $\alpha_l \leftarrow \alpha_{l+1} * \Phi_{l-1}^{C_{Y,U_j}} \bmod N^2$, for $2 \leq l \leq t+1$

  send A, Φ and α to each $V_i \in Ch(U_j)$

end if

{**Step 37 performed by** A **and** T}

$\widetilde{(RS}(A, U_j), U_j) \leftarrow \text{SMPA}(Z_j)$, for each Z_j

Algorithm 2 SMP$(E_{pk}(a), E_{pk}(b)) \rightarrow E_{pk}(a * b)$

Require: A has $E_{pk}(a)$ and $E_{pk}(b)$

1: A:

(a). Pick two random numbers $r_a, r_b \in \mathbb{Z}_N$
(b). $z_a \leftarrow E_{pk}(a) * E_{pk}(r_a) \bmod N^2$
(c). $z_b \leftarrow E_{pk}(b) * E_{pk}(r_b) \bmod N^2$; send z_a, z_b to T

2: T:

(a). Receive z_a and z_b from A
(b). $u_a \leftarrow D_{pr}(z_a)$; $u_b \leftarrow D_{pr}(z_b)$
(c). Compute $u = u_a * u_b \bmod N$
(d). $v \leftarrow E_{pk}(u)$; send v to A

3: A:

(a). Receive v from T
(b). $s \leftarrow v * E_{pk}(a)^{N-r_b} \bmod N^2$
(c). $s' \leftarrow s * E_{pk}(b)^{N-r_a} \bmod N^2$
(d). $E_{pk}(a * b) \leftarrow s' * E_{pk}(r_a * r_b)^{N-1} \bmod N^2$

$$\Phi_l = E_{pk}(M'_{l+1})$$

where M'_{l+1} is the scalar factor for users at $l+1$ hops away from A.

Proof For $h = 2$, we have $M'_2 = 1$ and it is clear that $\Phi_1 = E_{pk}(1) = E_{pk}(M'_2)$. Note that irrespective of the value of h, we always have $\Phi_{h-1} = E_{pk}(1) = E_{pk}(M'_h)$. When $h \geq 3$, initially the internal user X with $t = 2$ (denoting level $h-2$) sends $L_X = E_{pk}(\sum_{i=1}^{|Ch(X)|} C_{X,Y_i})$ to Z, where $Y_i \in Ch(X)$ and $Z = Pr(X)$. Then, Z aggregates the data received from $Ch(Z)$. Without loss of generality, let Z receives $L_{X_1}, \ldots, L_{X_d}$, where $X_i \in Ch(Z)$. Then, the aggregated entry in L_Z is $L_Z[1] = L_{X_1}[1] * \ldots * L_{X_d}$. In addition, Z sets $L_Z[2] = E_{pk}(\sum_{i=1}^{|Ch(Z)|} C_{Z,X_i})$. Since we are aggregating data component-wise, lth component in L_Z is equivalent to the encryption of summation of (minimum) number of messages exchanged between users at $L(h-l-1)$ and $L(h-l)$ under sub-tree of Z. (Note that, following from Observation 2, if X_i has multiple parents, then he/she will send L_{X_i} to only actual parent user $Pr(X_i)$). This aggregation process continues at each level in a bottom-up fashion. Finally, when A computes L_A (by aggregating the L_{B_i}'s component-wise, for $1 \leq i \leq m$), we observe that the lth component in L_A is equivalent to the encryption of sum of (minimum) number of messages exchanged between users at $L(h-l-1)$ and $L(h-l)$, that is, $L_A[l] = E_{pk}(M_{h-l-1,h-l})$, for $1 \leq l \leq h-2$. As mentioned earlier, let $L_A = \langle E_{pk}(x_1), \ldots, E_{pk}(x_{h-2}) \rangle$, where $x_l = M_{h-l-1,h-l}$, for $1 \leq l \leq h-2$. Based on the above discussions, we consider the following two scenarios depending on the value of h:

Scenario 1: When $h = 3$, we have $|L_A| = 1$ and Φ_1 gives the encrypted scalar factor for users at level 2 as shown below.

$$\begin{aligned}\Phi_1 &= L_A[1] \\ &= E_{pk}(M_{1,2}) \\ &= E_{pk}\left(\frac{M_{0,1} * M_{1,2}}{M_{0,1}}\right) \\ &= E_{pk}\left(\frac{M_3}{M_2}\right)\end{aligned}$$

Scenario 2: On the other hand, when $h > 3$, A and T jointly involve in the SMP protocol (Step 14 in Algorithm 1). Following from the SMP protocol (as given in Algorithm 2), the value of Φ_l, for $1 \leq l \leq h-2$, can be formulated as shown below:

$$\begin{aligned}\Phi_l &= E_{pk}\left(\prod_{j=1}^{h-l-1} x_j\right) \\ &= E_{pk}\left(\prod_{j=1}^{h-l-1} M_{h-j-1,h-j}\right) \\ &= E_{pk}\left(\prod_{k=l}^{h-2} M_{k,k+1}\right) \\ &= E_{pk}\left(\frac{M_{0,1} * \ldots * M_{h-2,h-1}}{M_{0,1} * \ldots * M_{l-1,l}}\right) \\ &= E_{pk}\left(\frac{M_h}{M_{l+1}}\right) \\ &= E_{pk}(M'_{l+1})\end{aligned}$$

□

4.2 Phase 2: Secure Computation of Recommendation Scores

During Phase 2, A with input Φ along with T and the internal users jointly compute the recommendation score for each potential candidate. The overall steps involved in Phase 2 of the PFR protocol are shown as steps 17–37 in Algorithm 1. To start with, initially A computes a vector α (which is different for each B_i) of size h as follows:

$$\begin{aligned}\alpha_1 &= E_{pk}(C_{A,B_i}) \\ \alpha_l &= \Phi_{l-1}^{C_{A,B_i}} \bmod N^2, \text{ for } 2 \leq l \leq h\end{aligned}$$

After this, A sends A, Φ, and corresponding α to B_i, for $1 \leq i \leq m$. Then, each internal user U_j receives the values of A, Φ, and α from $Pr(U_j)$ and checks whether

A is already a friend of U_j (this case happens only if U_j is equal to one of the B_i's). If $A \in F(U_j)$, then U_j simply forwards A, Φ, and α to each of his/her child friend. Otherwise, U_j computes the encryptions of parts of his/her recommendation score as below:

$$\beta_j = \alpha_1^{D_{U_j}} \bmod N^2;\ \gamma_j = \alpha_2 * \Phi_1^{C_{Y,U_j}} \bmod N^2$$

where D_{U_j} denote the degree of U_j (i.e., $|F(U_j)|$); Y denote the parent friend of U_j; β_j and γ_j denote the encryption of $D_{U_j} * \prod_i C_{S_{i-1},S_i}$ and $M'_l * |P(A,U_j)|$ respectively. Observe that $\widetilde{RS}(A,U_j) = M'_l * \left(|P(A,U_j)| * \prod_i C_{S_{i-1},S_i}\right) * D_{U_j}$. After this, U_j sends $Z_j = \{E_{pk}(U_j), \langle \beta_j, \gamma_j \rangle\}$ to A. Note that U_j can receive multiple pairs of (Φ, α) which occurs only when there exist multiple shortest paths from A to U_j. Under this scenario, U_j creates the encrypted partial scores for each pair of (Φ, α) and simply appends them to Z_j as follows.

$$Z_j = \{E_{pk}(U_j), \langle \beta_{1,j}, \gamma_{1,j} \rangle, \ldots, \beta_{s,j}, \gamma_{s,j} \rangle\}$$

where each $\beta_{l,j}, \gamma_{l,j}$, for $1 \le l \le s$, is computed as explained above for each pair of (Φ, α) and s denotes the number of such pairs (number of shortest paths to U_j from A). In addition, if the counter (t) corresponding to U_j is greater than 0, then U_j generates necessary information for his/her child friends as follows.

- Update Φ and α:
  - $\Phi_l = \Phi_{l+1}$, for $1 \le l \le t$
  - $\alpha_1 = \alpha_1^{C_{Y,U_j}} \bmod N^2$
  - $\alpha_l = \alpha_{l+1} * \Phi_{l-1}^{C_{Y,U_j}}$, for $2 \le l \le t+1$
- Send A, Φ and α to his/her child friends. If U_j receives multiple pairs of (Φ, α), U_j updates each pair as above and sends all updated pairs to the child friends.

Upon receiving the entries from all potential candidates, A and T involve in a secure multiplication and addition (SMPA) protocol. The main steps involved in the SMPA protocol are shown in Algorithm 3. Without loss of generality, consider the entry $Z_j = \{E_{pk}(U_j), \langle \beta_{1,j}, \gamma_{1,j} \rangle, \ldots, \langle \beta_{s,j}, \gamma_{s,j} \rangle\}$, where s denote the number of shortest paths from A to U_j. In addition, let $\beta_{k,j} = E_{pk}(a_{k,j})$ and $\gamma_{k,j} = E_{pk}(b_{k,j})$, for $1 \le k \le s$. The goal of the SMPA protocol is to securely compute $a_{1,j} * b_{1,j} + \cdots + a_{s,j} * b_{s,j}$ as output without revealing the values of $a_{k,j}$ and $b_{k,j}$, for $1 \le k \le s$, to either A or T. At the end of the SMPA protocol, only user A knows the recommendation score corresponding to U_j, for $1 \le j \le n$. The basic idea of the SMPA protocol is based on the following property which holds for any given $a_{k,j}, b_{k,j} \in \mathbb{Z}_N$, for $1 \le k \le s$:

Algorithm 3 SMPA

Require: A's input is Z_j

1: A:

(a). **for** $1 \leq k \leq s$ **do**:

- $\widetilde{\beta}_{k,j} \leftarrow \beta_{k,j} * E_{pk}(r_{k,j}) \bmod N^2$, where $r_{k,j} \in \mathbb{Z}_N$
- $\widetilde{\gamma}_{k,j} \leftarrow \gamma_{k,j} * E_{pk}(r'_{k,j}) \bmod N^2$, where $r'_{k,j} \in \mathbb{Z}_N$

(b). $\lambda_j \leftarrow E_{pk}(U_j) * E_{pk}(r_j) \bmod N^2$, where $r_j \in \mathbb{Z}_N$

(c). $E_{pk}(r) \leftarrow E_{pk}(\sum_{k=1}^{s} r_{k,j} * r'_{k,j})$

(d). $E_{pk}(r_1) \leftarrow \prod_{k=1}^{s} \beta_{k,j}^{r'_{k,j}} \bmod N^2$

(e). $E_{pk}(r_2) \leftarrow \prod_{k=1}^{s} \gamma_{k,j}^{r_{k,j}} \bmod N^2$

(f). $\tau \leftarrow E_{pk}(\tilde{r}_j) * E_{pk}(r)^{N-1} \bmod N^2$, where $\tilde{r}_j \in \mathbb{Z}_N$

(g). $w_j = \tau * E_{pk}(r_1)^{N-1} * E_{pk}(r_2)^{N-1} \bmod N^2$

(h). Send w_j, λ_j and $\widetilde{\beta}_{k,j}$, $\widetilde{\gamma}_{k,j}$, for $1 \leq k \leq s$ to T

2: T:

(a). Receive parameters from A

(b). $\widetilde{a}_{k,j} \leftarrow D_{pr}(\widetilde{\beta}_{k,j})$; $\widetilde{b}_{k,j} \leftarrow D_{pr}(\widetilde{\gamma}_{k,j})$, for $1 \leq k \leq s$

(c). $c_j \leftarrow \sum_{k=1}^{s} \widetilde{a}_{k,j} * \widetilde{b}_{k,j} \bmod N$

(d). $z_j \leftarrow D_{pr}(w_j)$; $s_{1,j} \leftarrow z_j + c_j \bmod N$

(e). $s_{2,j} \leftarrow D_{pr}(\lambda_j)$; send $s_{1,j}$ and $s_{2,j}$ to A

3: A:

(a). Receive $s_{1,j}$ and $s_{2,j}$ from T

(b). $\widetilde{RS}(A, U_j) \leftarrow s_{1,j} - \tilde{r}_j \bmod N$ (recommendation score)

(c). $U_j \leftarrow s_{2,j} - r_j \bmod N$ (corresponding user ID)

$$\sum_{k=1}^{s} a_{k,j} * b_{k,j} = \sum_{k=1}^{s} (a_{k,j} + r_{k,j}) * (b_{k,j} + r'_{k,j}) - \sum_{k=1}^{s} a_{k,j} * r'_{k,j} - \sum_{k=1}^{s} b_{k,j} * r_{k,j} - \sum_{k=1}^{s} r_{k,j} * r'_{k,j}$$

where $r_{k,j}$ and $r'_{k,j}$ are random numbers in $\mathbb{Z}_N$ and all arithmetic operations are performed under modulo N. The overall steps involved in SMPA are shown in Algorithm 3. Initially, A randomizes each encrypted tuple $\langle \beta_{k,j}, \gamma_{k,j} \rangle$, for $1 \leq k \leq s$, as follows:

$$\widetilde{\beta}_{k,j} = \beta_{k,j} * E_{pk}(r_{k,j}) \bmod N^2$$
$$\widetilde{\gamma}_{k,j} = \gamma_{k,j} * E_{pk}(r'_{k,j}) \bmod N^2$$

Here $r_{k,j}$ and $r'_{k,j}$ are randomly chosen in $\mathbb{Z}_N$. A also randomizes $E_{pk}(U_j)$ and performs these homomorphic operations (steps 1(b) to 1(g) of Algorithm 3). The r_j and $\tilde{r}_j$ are also random numbers in $\mathbb{Z}_N$. Then, A sends $\widetilde{\beta}_{k,j}$ and $\widetilde{\gamma}_{k,j}$, for $1 \leq k \leq s$, to T along with w_j and λ_j. Upon receiving, T decrypts $\widetilde{\beta}_{k,j}$ and $\widetilde{\gamma}_{k,j}$, for $1 \leq k \leq s$, multiplies and adds them as below:

- For $1 \leq k \leq s$, $\widetilde{a}_{k,j} = D_{pr}(\widetilde{\beta}_{k,j})$ and $\widetilde{b}_{k,j} = D_{pr}(\widetilde{\gamma}_{k,j})$
- $c_j = \sum_{k=1}^{s} \widetilde{a}_{k,j} * \widetilde{b}_{k,j} \bmod N$.

Furthermore, T decrypts w_j and λ_j: $z_j = D_{pr}(w_j)$ and $s_{2,j} = D_{pr}(\lambda_j)$, and computes $s_{1,j} = z_j + c_j \bmod N$. Then, T sends $s_{1,j}$ and $s_{2,j}$ to A. Finally, A removes the randomness from $s_{1,j}$ and $s_{2,j}$ to get the actual score and user ID U_j as follows:

$$\widetilde{RS}(A, U_j) = s_{1,j} - \tilde{r}_j \bmod N;\ U_j = s_{2,j} - r_j \bmod N$$

Here, $\widetilde{RS}(A, U_j)$ is the recommendation score for user U_j based on Eq. 5. Note that $(N-1)$ represents "-1" under $\mathbb{Z}_N$.

Theorem 2 *The output of Phase 2 is the list of recommendation scores along with the corresponding users IDs. That is, for any given entry Z_j, we have:*

$$s_{1,j} - \tilde{r}_j \bmod N = \widetilde{RS}(A, U_j)$$
$$s_{2,j} - r_j \bmod N = U_j$$

where $s_{1,j}$ and $s_{2,j}$ are the final values sent to A from T corresponding to the entry Z_j in the SMPA protocol, for $1 \leq j \leq n$.

Proof Without loss of generality, consider a potential user U_j who receives A and (Φ, α) pairs from his/her parent friends. Let us assume that U_j receives s number of different (Φ, α) pairs (representing s number of shortest paths from A to U_j) and let $\beta_{k,j}, \gamma_{k,j}$ denote the encrypted partial scores corresponding to kth pair (Φ_k, α_k) (denoting kth shortest path from A to U_j), for $1 \leq k \leq s$. U_j computes the encryptions of parts of kth pair as follows:

$$\beta_{k,j} = \alpha_{1,k}^{D_{U_j}} \bmod N^2 = E_{pk}\left(D_{U_j} * \prod_i C_{S_{i-1},S_i}\right)$$
$$\gamma_{k,j} = \alpha_{2,k} * \Phi_{1,k}^{C_{Y,U_j}} \bmod N^2 = E_{pk}(M'_l * |P_k(A, U_j)|)$$

where $\alpha_{y,k}$ (resp., $\Phi_{y,k}$) denotes the yth component of vector α_k (resp., Φ_k); $i = 1, \ldots, l-1$; $l = L(U_j)$ and $S_{i-1} = Pr(S_i)$ along the kth path from A to U_j. Then, U_j sends $Z_j = \{E_{pk}(U_j), \langle \beta_{1,j}, \gamma_{1,j} \rangle, \ldots, \langle \beta_{s,j}, \gamma_{s,j} \rangle\}$ to A. Upon receiving, A and T involve in the SMPA protocol. As mentioned earlier, let $\beta_{k,j} = E_{pk}(a_{k,j})$ and $\gamma_{k,j} = E_{pk}(b_{k,j})$, for $1 \leq k \leq s$. Since the SMPA protocol securely multiplies each $(\beta_{k,j}, \gamma_{k,j})$ pair and then adds them, the output of the SMPA protocol can be formulated as follows:

$$
\begin{aligned}
s_{1,j} - \tilde{r}_j \bmod N &= \sum_{k=1}^{s} a_{k,j} * b_{k,j} \\
&= \sum_{k=1}^{s} (D_{U_j} * \prod_i C_{S_{i-1},S_i}) * (M'_l * |P_k(A, U_j)|) \\
&= M'_l * \sum_{k=1}^{s} \left(|P_k(A, U_j)| * \prod_i C_{S_{i-1},S_i} \right) * D_{U_j} \\
&= \widetilde{RS}(A, U_j)
\end{aligned}
$$

Similarly, we can show that $s_{2,j} - r_j \bmod N = U_j$. □

During the actual implementation, the SMPA protocol can be initiated in parallel as the computation for potential user U_j is independent of others. Thus, overall, SMPA requires only one round of communication between A and T.

Example 3 We show various intermediate steps and results involved in the PFR protocol using Fig. 2 as an example. We have $h = 3$ and Lee as the target user. Following from initialization step, users at 1-hop away from Lee, that is, $\langle Hall, Cox, Bell \rangle$ have a value of $t = 2$. Similarly, $\langle Ford, Butler, Cole, Kelly \rangle$ have a value of $t = 1$. Whereas, $\langle Shaw, Ray, Fox, Ryan, Hart, Jones \rangle$ have $t = 0$. Each of them is aware of pk and also their parent and child friends (following from the initialization step).

Phase 1: Initially, $Hall$ computes $L_{Hall}[1] = E_{pk}(C_{Hall,Ford} + C_{Hall,Butler}) = E_{pk}(3)$. Similarly, Cox and $Bell$ compute $L_{Cox}[1] = E_{pk}(7)$ and $L_{Bell}[1] = E_{pk}(3)$ respectively. Observe that $C_{Hall,Cox}$ is not included in $L_{Hall}[1]$ and $L_{Cox}[1]$ since $Hall$ and Cox are at the same level from Lee. After this, $Hall, Cox$, and $Bell$ send L_{Hall}, L_{Cox}, and L_{Bell} resp., to Lee. Upon receiving values, Lee computes $L_{Lee}[1] = L_{Hall}[1] * L_{Cox}[1] * L_{Bell}[1] \bmod N^2 = E_{pk}(13)$. Then, Lee sets the encrypted scalar factors as follows:

$$\Phi_1 = L_{Lee}[1] = E_{pk}(13);\ \Phi_2 = E_{pk}(1)$$

Phase 2: During Phase 2, Lee computes encrypted vector α (different) for each of his friends. Without loss of generality, consider user $Hall$. Lee creates α for $Hall$ as follows.

$$
\begin{aligned}
\alpha &= \langle E_{pk}(C_{Lee,Hall}), \Phi_1^{C_{Lee,Hall}}, \Phi_2^{C_{Lee,Hall}} \rangle \\
&= \langle E_{pk}(2), E_{pk}(2 * 13), E_{pk}(2 * 1) \rangle
\end{aligned}
$$

Then, Lee sends $\langle Lee, \Phi, \alpha \rangle$ to $Hall$ who further forwards them to Ford and Butler. The final entries (that are sent to Lee) from all potential users are shown in Table 2. Finally, Lee and T involve in the SMPA protocol to get the scaled recommendation scores. E.g., the recommendation score for $Ford$ is $\widetilde{RS}(Lee, Ford) = 2 * D_{Ford} * 4 * 13 = 104 * D_{Ford}$. It is clear that $\widetilde{RS}(Lee, Ford) = M_h * TN * RS$, where actual

Table 2 Encrypted partial scores corresponding to each potential candidate based on PFR

$\{E_{pk}(Butler), \langle E_{pk}(2 * D_{Butler}), E_{pk}(39)\rangle, \langle E_{pk}(2 * D_{Butler}), E_{pk}(78)\rangle\}$
$\{E_{pk}(Fox), \langle E_{pk}(2 * D_{Fox}), E_{pk}(7)\rangle, \langle E_{pk}(6 * D_{Fox}), E_{pk}(10)\rangle\}$
$\{E_{pk}(Ryan), \langle E_{pk}(2 * D_{Ryan}), E_{pk}(5)\rangle, \langle E_{pk}(6 * D_{Ryan}), E_{pk}(8)\rangle\}$
$\{E_{pk}(Hart), \langle E_{pk}(6 * D_{Hart}), E_{pk}(10)\rangle, \langle E_{pk}(3 * D_{Hart}), E_{pk}(6)\rangle\}$
$\{E_{pk}(Ray), \langle E_{pk}(4 * D_{Ray}), E_{pk}(5)\rangle\}$
$\{E_{pk}(Jones), \langle E_{pk}(3 * D_{Jones}), E_{pk}(8)\rangle\}$
$\{E_{pk}(Ford), \langle E_{pk}(2 * D_{Ford}), E_{pk}(52)\rangle\}$
$\{E_{pk}(Cole), \langle E_{pk}(2 * D_{Cole}), E_{pk}(65)\rangle\}$
$\{E_{pk}(Kelly), \langle E_{pk}(D_{Kelly}), E_{pk}(52)\rangle\}$
$\{E_{pk}(Shaw), \langle E_{pk}(4 * D_{Shaw}), E_{pk}(6)\rangle\}$

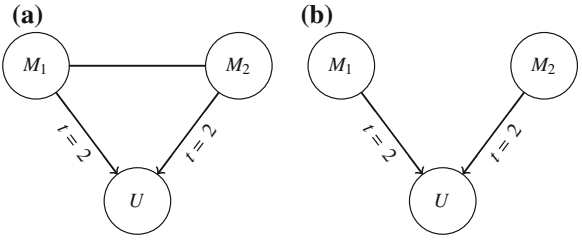


Fig. 3 **a** M_1 and M_2 are friends. **b** M_1 and M_2 are not friends. Case 1 of initialization step in PFR

recommendation score for $Ford$ is $RS = 4 * \frac{2}{5} * \frac{D_{Ford}}{TN}$ and $M_h = 65$. Note that, upon receiving α from $Hall$, $Ford$ computes β_{Ford} locally using D_{Ford} (known only to $Ford$) and homomorphic multiplication as $\beta_{Ford} = \alpha_1^{D_{Ford}} = E_{pk}(2 * D_{Ford})$. □

4.3 Security Analysis

In this sub-section, we analyze the security of the initialization step and each Phase in the proposed PFR protocol separately.

We emphasize that, during the initialization step, the generation of candidate network does not leak any information. Consider the two possible cases of the initialization step as discussed in Observation 2. For case 1, where U receives the same counter values from multiple parents (on the same level), U cannot predict whether there exist friendship between any two parents. For example, suppose U receives $t = 2$ from two parents M_1 and M_2. Then we observe that U cannot distinguish the two scenarios shown in Fig. 3. This is because the value of t received by U is independent of the relationship between the parents who sent it.

Similarly, for case 2, where U can receive different counter values from multiple parents (different levels), U cannot deduce any information by simply using the received counter values. For example, consider that U receives $t = 3$ from M_1 and

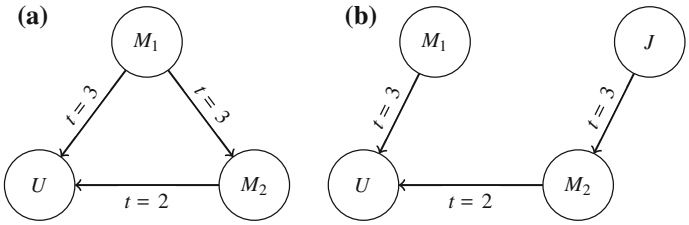


Fig. 4 **a** M_1 and M_2 are friends. **b** M_1 and M_2 are not friends. Case 2 of initialization step in PFR

$t = 2$ from M_2. Then, U cannot distinguish between the following two possible scenarios as shown in Fig. 4, where J is the parent user of M_2. It is clear that the counter values passed to U are independent of the relationship between M_1 and M_2. Hence, passing the counter values during the formation of candidate network (initialization step) in PFR does not reveal any information.

During Phase 1, each internal user sends the encrypted aggregated data only to $Pr(U)$. Thus, the privacy of individual users is preserved as per the security definition of SMC [17]. In addition, during the SMP protocol, A first randomizes the values of L_A and sends them to T. Therefore, the simulated view of T is indistinguishable compared to the real view (trusted third party model). Furthermore, since T sends only the encrypted scalar factors to A, neither the values of $M_{i-1,i}$'s nor M'_i's are revealed to A, for $1 \leq i \leq h-1$. Therefore, the privacy of A and U_j are preserved, for $1 \leq j \leq n$. Note that the scalar factor for users at level h is always known to A, since $M'_h = 1$ always hold. However, we emphasize that this does not reveal any information to A.

On the other hand, in Phase 2, A initially sends $\{A, \Phi, \alpha\}$ to each B_i, for $1 \leq i \leq m$. Each internal user U_j, computes his/her encrypted partial scores using D_{U_j} and C_{Y_i,U_j}, where Y_i is the parent friend of U_j (with multiple parents denoting multiple shortest paths from A to U_j). Then, U_j sends his entry Z_j in encrypted form to A. Here, the privacy of each U_j is preserved under the assumption that number of shortest paths to U_j can be revealed to A. However, we emphasize that this problem can be solved my random masking without effecting the recommendation score (more details are given in Sect. 6). During the SMPA protocol, the values of each entry are randomized in $\mathbb{Z}_N$ and sent to T. That is, the values of $D_{U_j} * \prod_i C_{S_{i-1},S_i}$ and $M'_l * |P_k(A, U_j)|$, for $1 \leq k \leq s$, are randomized and sent to T. Therefore, the privacy of A and U_j is preserved further. In addition, the final output sent to A is the actual output and the intermediate values are never revealed to A, T and other internal users.

Based on the above discussions, it is clear that, apart from the initialization step, Phase 1 and 2 are secure. In addition, the values returned from Phase 1 to 2 are pseudo-random; therefore, the sequential composition of the two Phases lead to a secure protocol according to the Composition Theorem [18].

4.4 Complexity Analysis

We analyze the computation and communication costs of each party in the PFR protocol. For the rest of this sub-section we omit the costs of the Initialization step since they are negligible compared to the encryption costs in Phase 1 and 2 of PFR.

4.4.1 Computation Cost

For Phase 1, the computation cost of each internal user U_j depends on his/her counter value and number of child friends. In addition, irrespective of the counter value, U_j has to perform one encryption operation. Therefore, the computation complexity of U_j is bounded by one encryption and $O(t * |Ch(U_j)|)$ homomorphic addition operations. Whereas, the computation complexity of A mainly depends on h and m. If $h = 2$, then A simply performs one encryption operation. However, when $h = 3$, A's computation complexity is bounded by $O(h * m)$ homomorphic additions and one encryption. On the other hand, if $h > 3$, the computation complexity of A mainly comes from the SMP protocol which depends on the value of h. That is, A's computation complexity is bounded by $O(h * m)$ homomorphic additions and $O(h)$ number of encryption operations. Whereas, the computation complexity of T is bounded by $O(h)$ decryption operations (coming from the SMP protocol).

In Phase 2, the computation complexity of each internal user (excluding B_i's) depends on his/her t and s (number of shortest paths from A to U_j). Specifically, U_j's computation cost is bounded by $O(t * s)$ exponentiations and homomorphic additions. On the other hand, A has to initially compute α, which depends on the value of h, for each B_i. Therefore, the computation complexity of A for computing all α values is bounded by $O(h * m)$ encryption and exponentiation operations. In addition, during the SMPA protocol, A has to randomize all components of each potential candidate. Let n denote the number of potential candidates and s be the maximum number of shortest paths, then the computation cost of A in the SMPA protocol is bounded by $O(s * n)$ encryption and exponentiation operations. Overall, during Phase 2, the computation complexity of A is bounded by $O(s * n)$ encryption and exponentiation operations (under the assumption $s * n > h * m$). Whereas, the computation complexity of T is bounded by $O(s * n)$ decryption operations (coming from the SMPA protocol).

4.4.2 Communication Cost

Without loss of generality, let K denote the Paillier encryption key size (in practice, K should be at least 1,024 bits). During Phase 1, the communication complexity between any two internal users is bounded by $O(K * t)$ bits. Note that t may vary between each pair of users depending on their location in the corresponding candidate network and only adjacent users (i.e., friends) communicate to each other.

Whereas, between A and all B_i's, the communication cost is bounded by $O(K * h * m)$ bits. In addition, for the SMP protocol, the communication complexity between A and T is bounded by $O(K * h)$ bits.

Additionally, during Phase 2, the communication cost between any two internal users is bounded by $O(K * t)$ bits, where t is the counter of the corresponding parent user. Since A has to send Φ and α to each B_i, for $1 \leq i \leq m$, the communication cost between A and all B_i's is bounded by $O(K * h * m)$ bits. In addition, since each potential candidate sends the encryptions of parts of his/her recommendation score to A, the communication cost between A and all potential candidates is bounded by $O(K * s * n)$ bits. Here, we assume there exist s number of shortest paths from A to each potential candidate. Finally, during the SMPA protocol, the communication cost between A and T is bounded by $O(K * s * n)$ bits.

5 Practical Implementation Details

5.1 Masking Number of Shortest Paths

As mentioned earlier, the PFR protocol reveals (only to A) the number of shortest paths from A to each potential candidate U_j, for $1 \leq j \leq n$. However, it is unclear how this additional information effects the privacy of U_j. Nevertheless, we emphasize that this problem can be solved by randomly masking the number of entries corresponding to each U_j without effecting his/her final recommendation score. Suppose $Z_j = \{E_{pk}(U_j), \langle \beta_{1,j}, \gamma_{1,j} \rangle, \ldots, \langle \beta_{s,j}, \gamma_{s,j} \rangle\}$ is the entry corresponding to U_j, where s denotes the number of shortest paths from A to U_j in the corresponding candidate network. We briefly explain the steps involved in masking the entry Z_j by U_j below, for $1 \leq j \leq n$:

- Randomly mask the number of shortest paths from A to U_j by computing $Z'_j = \{E_{pk}(U_j), \langle \beta_{1,j}, \gamma_{1,j} \rangle, \ldots, \beta_{s+\theta,j}, \gamma_{s+\theta,j} \rangle\}$, where θ is the security parameter chosen by U_j, such that $\beta_{s+i,j} = \gamma_{s+i,j} = E_{pk}(0)$, for $1 \leq i \leq \theta$. Note that the encryption scheme used in this chapter (based on HPEnc$^+$ system) is probabilistic. Therefore, encryption of 0 each time yields different (random) ciphertext in $\mathbb{Z}_{N^2}$.
- Send the masked entry Z'_j to A.

The rest of the steps are the same as in Phase 2 of PFR. Observe that applying SMPA on Z'_j yields the same result as on Z_j, for $1 \leq j \leq n$.

5.2 Data Encryption and Secure Peer-to-Peer Communication

The PFR protocol assumes there exist peer-to-peer network connectivity and the communication between any two users happens through a secure channel. However, in practice, as it is the case is many online social networks, communication between any two users should happen through the social network provider (say the network

administrator) for security reasons. Another reason for this is a user may not be online at the receiving end.

As mentioned in Sect. 1, in this chapter, we assume that users profile data are first encrypted (using his/her own secret key) and then stored on the server of network administrator [2]. We emphasize that this assumption is commonly made in the literature of related problem domains such as in [2, 7, 14, 26, 36, 37]. Also, from user's perspective, this assumption gives more flexibility to them since users have more control over their own data. In addition, secure communication between any two users can be achieved by establishing a secure session (using AES session key [31]) between them. For example, if Alice wants to establish a new secure session with Bob, then Alice encrypts the trapdoors using Bob's public key and sends it to the network administrator. Upon receiving, Bob can decrypt them to get the trapdoors which will enable Alice and Bob to communicate in a secure fashion. It is important to note that each user in the network is responsible to generate, maintain, and share his/her own keys.

Under such an architecture, the network administrator merely acts as an intermediate router who simply stores the encrypted data sent by the sender and delivers it to the concerned end-user after he/she logins. Note that the data to be sent is first encrypted using the corresponding session key of sender and then stored on the server. Therefore, only the end-user who holds the session key can decrypt it. For example, during the initialization step of PFR, each user U first establishes a secure session with his/her friends. Then, the value of t to be sent is encrypted and stored on the server. Once the intended end-users (i.e., friends of U) logins into social network, the network administrator sends the encrypted value of t to him/her who decrypts it to know his/her counter value. We emphasize that the session keys should be changed occasionally for security reasons.

6 Inherent Security Issue

In many online social networks, such as Facebook, we observe an inherent security issue due to the information flow between different entities. For example, consider the scenario of user U sending a message to another user V. Though the message is encrypted using the session key of U, as explained above, it is clear that the network administrator will know that U and V have some kind of relationship. This is because of the fact that the communication between any two users should pass through the network administrator. Here U and V might be friends but this may not be the case always since a user can send a message to any other user in the social network (i.e., no need of friendship between the two users).

In particular to the PFR protocol, this information might be too specific since U sends an encrypted counter or some other information during Phase 1 or 2 to only his/her parent or child nodes (i.e., friends of U). However, we emphasize that the network administrator cannot distinguish which message belongs to which application. This is because the messages are encrypted and thus the underlying application

is not transparent to the network administrator. Nevertheless, the network administrator will still know that there exist some kind of relationship between U and the users at the receiving end. It is unclear how this extra information will be useful to the network administrator in deducing any private information of U.

7 Extension to PFR

As mentioned above, the PFR protocol might leak the additional information that there exist some kind of relationship between U and his/her friends to the network administrator. Note that this kind of information is never revealed to other users in the PFR protocol. We emphasize that if the message flow is always between U and only his/her friends, then the probability of guessing user $V_j \in F(U)$ is a friend of U by the network administrator is very high. Therefore, to mitigate this issue or to provide better security, this section presents an extended version (denoted by PFR_{rand}) of the PFR protocol by randomly including additional users (apart from the actual friends) to take participation in the PFR protocol without effecting the final recommendation scores. By doing so, we are actually randomizing the friend lists of users; therefore, reducing the probability of guessing the friends of a user by the network administrator.

Similar to PFR, the PFR_{rand} protocol consists of an initialization step and two phases. However, the methodology is slightly different from PFR. Therefore, we explain the key steps in PFR_{rand} that are different from PFR below.

7.1 Initialization Step

To start with, the target user A selects a random set of dummy friends (excluding the actual friends of A) from the social network, denoted by $D(A)$, where $|D(A)|$ is the security parameter for A. Note that the dummy friends can also be referred to as dummy child nodes. Then, A sets the counter t to $h - 1$ and sends (δ, t) to each user Y_j in $F(A) \cup D(A)$. Where $\delta = 1$ if $Y_j \in F(A)$, and 0 otherwise. Then, each intermediate user U selects his/her random set of dummy users $D(U)$, stores (δ, t) and the parent user who sent the entry to U locally (as explained below). In addition, he/she sends the updated (δ, t) entry to users in $Ch(U) \cup D(U)$, where $Ch(U)$ denotes the actual child nodes of U as explained in Sect. 4. This process is continued until the users at h-hop away from A receive a counter of $t = 0$. Since U can receive multiple pairs of (δ, t), depending on whether U is a dummy user or not, we address the following two cases.

Case 1: If the δ values received by U are all 0's (whereas the values of t can be different), then U is actually a dummy user (i.e., not part of candidate network). Under this case, U stores δ as 0 and the maximum t value received locally. Plus, U

selects the user who sent the maximum t as his/her parent (i.e., $P(U)$). In addition, U sends (δ, t) to each user in $Ch(U) \cup D(U)$ with $\delta \leftarrow 0$ and $t \leftarrow t - 1$.

Case 2: On the other hand, if U receives either different δ values (i.e., both 0 and 1's) or same δ values of 1, then U is part of the candidate network. Therefore, U stores δ as 1 and selects the maximum t value among the entries with $\delta = 1$ as his/her counter value . Also, among the users who sent $\delta = 1$, U selects the user who sent the maximum t as $Pr(U)$. Unlike in the previous case, U sends $(1, t)$ to users in $Ch(U)$ and $(0, t)$ to users in $D(U)$, where $t \leftarrow t - 1$.

At the end of the initialization step, each user who participates in the PFR_{rand} protocol knows whether he/she is a dummy user (i.e., $\delta = 0$) or part of the candidate network ($\delta = 1$). In addition, each user knows his/her parent user ($Pr(U)$), actual child users ($Ch(U)$), and dummy child users ($D(U)$).

7.2 Phase 1: Secure Computation of Scalar Factors

Following from Case 1 and 2 of the above initialization step, we have the following observation in the PFR_{rand} protocol.

Observation 3. For any internal user U who belongs to the candidate network (i.e., $\delta = 1$), we observe that $Pr(U)$ also belongs to the candidate network and is the same as in the PFR protocol (assuming single parent).

The basic idea of Phase 1 in PFR_{rand} is the same as in PFR. However, the only difference is that whenever an internal node (and also A) receives the encrypted aggregated data from his/her child nodes (i.e., users in $Ch(U) \cup D(U)$), U simply dumps the messages received from dummy child nodes ($\in D(U)$). More specifically, only the messages received from actual child nodes are used in performing further computation by U and the resulting aggregated data is forwarded to $Pr(U)$. Since the messages from dummy nodes are not used for computations and following from Observation 3, we observe that the final output from Phase 1 is the same as that of Phase 1 in PFR.

7.3 Phase 2: Secure Computation of Recommendation Scores

Similar to PFR, at the end of Phase 1 in PFR_{rand}, A has the list of encrypted scalar factors (i.e., Φ) for all potential users within a radius of h. Note that, as mentioned above, these scalar factors are computed based on the candidate network. That is, even though dummy child friends are added for each user to provide better security, the corresponding messages are not used during the computation of encrypted scalar factors for each level. Thus, the final result of Phase 1 in PFR_{rand} is equivalent to Φ (list of encrypted scalar factors for each level).

During Phase 2, A initially sends his/her ID, Φ, and α (which is computed similar to in PFR) to each user in $F(A) \cup D(A)$. Then, each participating user M_j in PFR_{rand}, after receiving encrypted data from his/her parent(s), computes the encryptions of parts of his/her recommendation score as mentioned in PFR. After this, M_j sends his/her entry Z_j, as defined below, to A.

$$Z_j = \{E_{pk}(M_j), E_{pk}(\delta_j), \langle \beta_{1,j}, \gamma_{1,j} \rangle, \ldots, \beta_{s,j}, \gamma_{s,j} \rangle\}$$

where s denotes the number of shortest paths from A to M_j and the flag δ_j denotes whether or not M_j is part of the candidate network. Observe that δ_j is not used in the PFR protocol. We emphasize that, similar to PFR, the number of shortest paths can be masked by M_j before sending it to A. For each entry Z_j received, A first retrieves δ_j securely as follows:

- A randomizes $E_{pk}(\delta_j)$ by computing $\sigma_j = E_{pk}(\delta_j) * E_{pk}(r_j) \text{mod } N^2$, where r_j is a random number chosen from $\mathbb{Z}_N$, and sends σ_j to T.
- Upon receiving, T computes $\delta'_j = D_{pr}(\sigma_j)$ and sends δ'_j to A.
- A removes the randomization from δ'_j to get δ_j, i.e., computes $\delta_j = \delta'_j - r_j \text{ mod } N$.

At the end of this step, A knows which entries belong to actual potential users (i.e., $\delta_j = 1$) and dummy users (i.e., $\delta_j = 0$). Finally, A and T involve in the SMPA protocol to get the recommendation scores and corresponding user IDs for only those entries with $\delta_j = 1$. We emphasize that the number of instantiations of SMPA in PFR_{rand} is the same as in PFR.

8 Empirical Analysis

Since the effectiveness of PFR is the same as CSM method [9], in this section, we analyze the computation costs of PFR based on different parameters. In addition, we compare the computation costs of PFR_{rand} with PFR.

8.1 Platform and Dataset Description

For our experiments, we used Paillier cryptosystem [32] as the underlying encryption scheme. The proposed protocols were implemented in C, and experiments were conducted on an Intel® Xeon® Six-Core™ 3.07GHz PC with 12GB memory running Ubuntu 10.04 LTS.

Since it is hard to control parameters in a real-world dataset, we simulated the environment and computed the computation costs. In all our experiments, we assume that the minimum number of messages exchanged between any two users U and V (i.e., $C_{U,V}$) is uniformly distributed in [1, 1000]. Also, we assume that there is no

communication delay between participating users (which further implies that users are online). In addition, the number of child friends for each user (including the target user A) is varied from 50 to 250.

8.2 Performance of PFR

We first analyze the computation costs of A, T, and internal user U_j in Phase 1 and 2 of PFR separately. Since the run time of U_j depends on which level he/she belongs to, we present the average of computation costs of all internal users at different levels. For the rest of this sub-section, the Paillier's encryption key size (i.e., K) is fixed to either 1,024 or 2,048 bits.

For Phase 1, we fix the value of h to 6 and compute the run time of A, T, and U_j by varying the number of child friends from 50 to 250 (note that users at level $h-1$ and h do not involve in Phase 1). As shown in Fig. 5a, for Paillier key size $K = 1{,}024$ bits, the computation time for A, T, and U_j are 79, 30, and 4.25 ms respectively when the number of child friends is 50. In addition, the computation time of U_j varies only slightly (due to less expensive homomorphic operations) from 4.25 to 6 ms when the number of child friends of U_j are varied from 50 to 250. However, for A the computation time remains the same since the cost of homomorphic addition operations are negligible compared to the encryption operations involved in SMP. Since h is fixed, the encryption cost in SMP remains the same irrespective of the child friends of A. Therefore, the computation costs of A and T remains the same in Phase 1. A similar trend can be observed for $K = 2{,}048$ bits as shown in Fig. 5b. Briefly, when the number of child friends is 50, the computation costs of A, T, and U_j are 571, 240, and 24.25 ms respectively. Also, irrespective of the number of child friends and h, we observe that the computation time of U_j is always significantly less than that of A and T.

During Phase 2, the computation time to find the recommendation score for each potential candidate U_j mainly depends on m (number of A's friends) and the cost of SMPA which in turn depends on the number of shortest paths (s) from A to U_j. However, for any given m, the cost to compute recommendation score for each U_j varies with the corresponding s. Thus, we analyze the computation costs for A, T and U_j based on varying values of s with $h = 6$ and $m = 50$. As shown in Fig. 5c, the computation cost of U_j (averaged over different levels) increases from 0.8 to 2.4 ms when s is varied from 5 to 25 for $K = 1{,}024$ bits. However, due to the expensive encryption costs, the computation cost of A varies from 207 to 538 ms when s is changed from 5 to 25. On the other hand, due to the decryption costs in SMPA, the computation cost of T varies from 30 to 130 ms when s is changed from 5 to 25. A similar trend can be observed for $K = 2{,}048$ bits as shown in Fig. 5d. In short, when $s = 5$, the computation costs of A, T, and U_j are 1.79,0.24 and 0.002 s respectively. We observe that, for any fixed parameters, the computation costs of A and T are increased by a factor of almost 8 whereas U_j's time is increased by a factor of 2 when K is doubled.

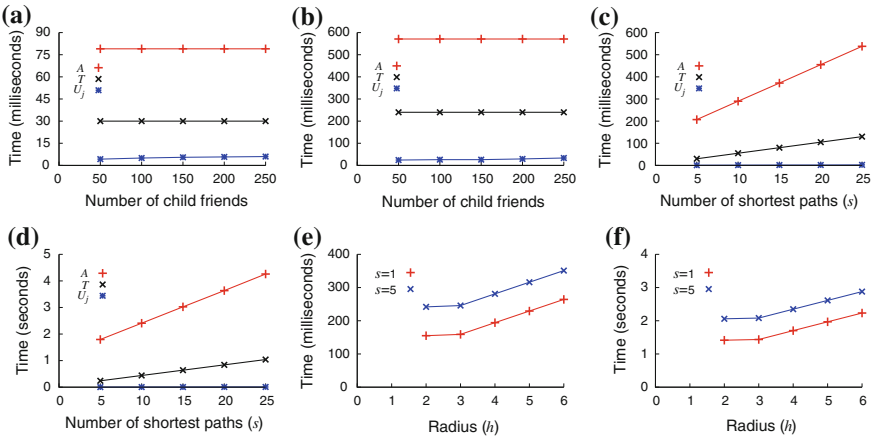


Fig. 5 **a** Phase 1 for $K = 1{,}024$. **b** Phase 1 for $K = 2{,}048$. **c** Phase 2 for $K = 1{,}024$. **d** Phase 2 for $K = 2{,}048$. **e** Total cost for $K = 1{,}024$. **f** Total cost for $K = 2{,}048$. Computation costs of different parties in PFR based on varying parameter values

Based on the above results, it is clear that the computation costs incurred due to Phase 2 are much higher than those of Phase 1. This further validates our computational analysis of PFR as discussed in Sect. 4.

Furthermore, we compute the total run time of PFR based on varying values of h and s. That is, the total time took by our PFR protocol to compute the recommendation score corresponding to the potential user U_j (similar analysis can be deduced for other potential candidates) is computed. We fix the number of child friends for each potential user (and A) to 50. The results are as shown in Figs. 5e and 5f. For K=1,024, as shown in Fig. 5e, we observe that the total time does not change much when h is changed from 2 to 3 for both s =1 and 5. For example, when $s = 5$, the run time of PFR varies from 241.8 to 245.8 ms when h is changed from 2 to 3. This is because the cost of Phase 1 does not change much since there is no need of SMP when $h = 2$ and 3. Also, the cost of Phase 2 almost remains the same for any fixed s (assuming m is also fixed). However, the total cost increases as h is varied from 3 to 6. For example, when $s = 5$, the total time for PFR to compute the recommendation score for U_j varies from 245.8 to 351.05 ms when the value of h is changed from 3 to 6. A similar trend can be observed for $s = 1$ as shown in Fig. 5e. Also, for any given value of h, the computation time of PFR is almost increased by a factor of 1 to 2 when s is changed from 1 to 5. E.g., when $h = 6$, the computation time of PFR varies from 264.25 to 351.05 ms when s is changed from 1 to 5. In addition, we observed that for any fixed values of h and s, the running time of PFR grows almost linearly with n. A similar trend can be observed for $K = 2{,}048$ as shown in Fig. 5f.

These results show that most of the significant computation (more than 97 %) is from A and T. The computation cost incurred on all internal nodes is negligible. In addition, for A and T the computation time grows linearly with s and n. Furthermore,

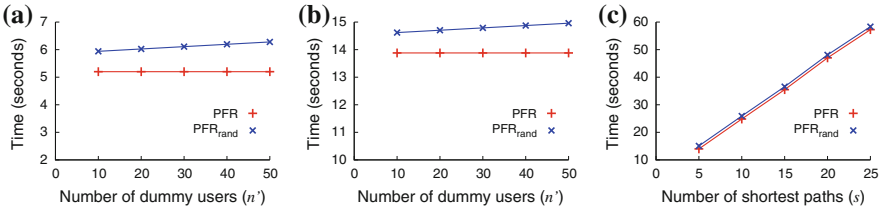


Fig. 6 **a** Cost of Phase 2 for $s = 1$. **b** Cost of Phase 2 for $s = 5$. **c** Total cost for $n' = 50$. Comparison of computation costs of PFR_{rand} with PFR for $K = 1,024$ bits

when the size of encryption key doubles, the computation time of PFR increases by almost a factor of 8 for any fixed parameters.

8.3 Computation Costs: PFR Versus PFR_{rand}

In this sub-section, we compare the computation costs of PFR_{rand} with PFR. As mentioned earlier, remember that adding random dummy users to take participation in PFR_{rand} does not change the relative ordering among the recommendation scores of actual potential candidates. That is, the effectiveness of PFR_{rand} is the same as in PFR. For the rest of this sub-section, we fix the Paillier key size K to 1,024 bits (however we emphasize that similar analysis can be deduced for $K = 2{,}048$ bits).

For any given parameters, it is important to note that the computation cost of Phase 1 in PFR_{rand} is the same as in PFR. This is because of the fact that though additional dummy child friends are added for each internal user (and A), he/she will operate on the encrypted partial data received from only his/her actual child friends. Therefore, we first compare the computation costs of Phase 2 in PFR and PFR_{rand}.

Suppose the number of actual child friends for each potential candidate U_j be 50, i.e., $|Ch(U_j)| = 50$. Also, let $m = 50$, $n = 100$ and $h = 6$. Now we evaluate the computation time of Phase 2 in the proposed protocols for varying n' and $s = 1$, where n' denotes the total number of dummy random users participating in the PFR_{rand} protocol. As shown in Fig. 6a, the computation time of Phase 2 in PFR is 5.2 s and it remains constant with varying n' since Phase 2 is independent of n' in PFR. Whereas, the computation cost of Phase 2 in PFR_{rand} varies from 5.93 to 6.27 s when n' is varied from 10 to 50. It is clear that the extra cost incurred on Phase 2 in PFR_{rand} is not that much compared to the cost of Phase 2 in PFR. A similar trend can be observed for $s = 5$ as shown in Fig. 6b. Briefly, the computation time of Phase 2 in PFR remains to be constant at 13.88 s whereas in PFR_{rand} the cost varies from 14.61 to 14.95 s when n' is changed from 10 to 50.

We have also computed the total run time of PFR and PFR_{rand} for varying s and by fixing n' to 50. The results are as shown in Fig. 6c. The total run time of PFR varies from 13.99 to 57.25 s whereas that of PFR_{rand} varies from 15.07 to 58.33 s when

s is changed from 5 to 25. Irrespective of the value of s and for any fixed parameters, we observe that the total run time of PFR is very close to that of PFR_{rand}.

We emphasize that friend recommendation is generally performed on a daily basis (not a real-time application); therefore, the performances of the proposed protocols are reasonable in practice. The main advantage is that the proposed protocols make it possible to do friend recommendations even in a privacy-preserving social networking environments.

9 Conclusion

Friend recommendation has become an important service in many online social networks to enhance the development of the entire network as well as to provide opportunities for users to expand their social relations. Due to privacy concerns [10, 21, 40], many online social networks are providing various privacy settings to users. Existing friend recommendation algorithms do not take privacy into account; therefore, they are not applicable in privacy-preserving social networking environments. In this chapter, we first proposed a new two-phase private friend recommendation (PFR) algorithm based on the network structure as well as real messages exchanged between users. The proposed PFR protocol computes the recommendation scores of all users within a radius of h from the target user A by using the similarity metric proposed in [9] as a baseline. In particular, our protocol generates the (scaled) recommendation scores along with the corresponding user IDs in such a way that the relative ordering among the users in the TOP-K list of recommended users is preserved (i.e., the same accuracy as in [9]).

We have provided an in-depth security and complexity analysis of the proposed PFR protocol and also addressed various implementation details related to PFR in practice. In addition, we demonstrated a new security issue in the current online social networks due to the inherent message flow information between different entities. To mitigate this issue or to provide better security, we proposed an extended version of the proposed protocol using randomization technique. We also showed the practical applicability of the proposed protocols through extensive experiments based on different parameters.

Acknowledgments This material is based upon work supported by the Office of Naval Research under Award No. N000141110256 and NSF under award No. CNS-1011984.

References

1. Agrawal R, Srikant R (2000) Privacy preserving data mining. ACM SIGMOD 29:439–450
2. Baden R, Bender A, Spring N, Bhattacharjee B, Starin D (2009) Persona: an online social network with user-defined privacy. ACM SIGCOMM Comput Commun Rev 39(4):135–146
3. Bonchi F, Castillo C, Gionis A, Jaimes A (2011) Social network analysis and mining for business applications. ACM Trans Intell Syst Technol 2:22:1–22:37

4. Boyd D, Ellison NB (2008) Social network sites: definition, history, and scholarship. J Comput-Mediated Commun 13(1):210–230
5. Chen J, Geyer W, Dugan C, Muller M, Guy I (2009) Make new friends, but keep the old: recommending people on social networking sites. In: Proceedings of the 27th international conference on Human factors in computing systems, pp 201–210
6. Clarkson MR, Chong S, Myers A (2008) Civitas: toward a secure voting system. In: IEEE symposium on security and privacy, pp 354–368, May 2008
7. Cristofaro ED, Soriente C, Tsudik G, Williams A (2012) Hummingbird: privacy at the time of twitter. In: IEEE symposium on security and privacy, pp 285–299
8. Cutillo LA, Molva R, Onen M (2011) Analysis of privacy in online social networks from the graph theory perspective. In: Proceedings of the global telecommunications conference (GLOBECOM 2011). IEEE, Houston, Texas, USA, pp 1–5, Dec 2011
9. Dai B-R, Lee C-Y, Chung C-H (2011) A framework of recommendation system based on both network structure and messages. In: International conference on advances in social networks analysis and mining (ASONAM' 11), pp 709–714, July 2011
10. Delgado J, Rodríguez E, Llorente S (2010) User's privacy in applications provided through social networks. In: Proceedings of second ACM SIGMM workshop on social, media, pp 39–44
11. Dong W, Dave V, Qiu L, Zhang Y (2011) Secure friend discovery in mobile social networks. In: Proceedings of IEEE INFOCOM, pp 1647–1655, Apr 2011
12. Dwork C (2006) Differential privacy. In: ICALP, Lecture Notes in Computer Science, Springer, pp 1–12
13. Dwork C (2008) Differential privacy: a survey of results. In: Proceedings of the 5th international conference on theory and applications of models of computation, TAMC'08. Springer-Verlag, Berlin, Heidelberg, pp 1–19
14. Feldman AJ, Blankstein A, Freedman MJ, Felten EW (2012) Social networking with frientegrity: privacy and integrity with an untrusted provider. In: Proceedings of the 21st USENIX conference on security symposium, Security'12. Berkeley, CA, USA, pp 31–31. USENIX Association
15. Fontaine C, Galand F (2007) A survey of homomorphic encryption for nonspecialists. EURASIP J Inf Secur 2007:15:1–15:15
16. Gao H, Hu J, Huang T, Wang J, Chen Y (2011) Security issues in online social networks. IEEE Internet Comput 15(4):56–63
17. Goldreich O (2004) The foundations of cryptography, vol 2. In: General Cryptographic Protocols. Cambridge University Press, Cambridge
18. Goldreich O (2004) The foundations of cryptography, vol 2. In: Encryption Schemes. Cambridge University Press, Cambridge
19. Goldwasser S, Micali S, Rackoff C (1989) The knowledge complexity of interactive proof systems. SIAM J Comput 18:186–208
20. Gou L, You F, Guo J, Wu L, Zhang XL (2011) Sfviz: interest-based friends exploration and recommendation in social networks. In: Proceedings of visual information communication-international symposium (VINCI '11), ACM, pp 1–10
21. Krishnamurthy B, Wills CE (2010) On the leakage of personally identifiable information via online social networks. SIGCOMM Comput Commun Rev 40(1):112–117
22. Kumar R, Novak J, Tomkins A (2006) Structure and evolution of online social networks. In: The 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 611–617
23. Liben-Nowell D, Kleinberg J (2007) The link-prediction problem for social networks. J Am Soc Inf Sci Technol 58:1019–1031
24. Lindell Y, Pinkas B (2002) Privacy preserving data mining. J Cryptol 15:177–206
25. Lo S, Lin C (2006) Wmr-a graph-based algorithm for friend recommendation. In: Proceedings of the 2006 IEEE/WIC/ACM international conference on web intelligence (WI '06), IEEE Computer Society, pp 121–128
26. Lucas MM, Borisov N (2008) Flybynight: mitigating the privacy risks of social networking. In: Proceedings of the 7th ACM workshop on privacy in the electronic society, WPES '08. ACM, New York, NY, USA, pp 1–8

27. Machanavajjhala A, Korolova A, Sarma AD (2011) Personalized social recommendations: accurate or private. Proc VLDB Endowment 4:440–450
28. Milgram S (1967) The small world problem. Psychol Today 2:60–67
29. Mislove A, Marcon M, Gummadi KP, Druschel P, Bhattacharjee B (2007) Measurement and analysis of online social networks. In: Proceedings of the 7th ACM SIGCOMM conference on internet measurement (IMC '07), pp 29–42
30. Naruchitparames J, Giine M, Louis S (2011) Friend recommendations in social networks using genetic algorithms and network topology. In: IEEE congress on evolutionary computation (CEC), pp 2207–2214, June 2011
31. NIST. Advanced encryption standard. Technical Report NIST Special Publication FIPS-197, National Institute of Standards and Technology, 2001
32. Paillier P (1999) Public-key cryptosystems based on composite degree residuosity classes. In: Proceedings of the 17th international conference on theory and application of cryptographic techniques, Springer-Verlag, pp 223–238
33. Rivest RL, Adleman L, Dertouzos ML (1978) On data banks and privacy homomorphisms. In: Foundation of secure computations, Academic Press, pp 169–177
34. Silva N, Tsang I-R, Cavalcanti G, Tsang I-J (2010) A graph-based friend recommendation system using genetic algorithm. In: IEEE congress on evolutionary computation (CEC), pp 1–7, July 2010
35. Srivastava J, Ahmad MA, Pathak N, Hsu DKW (2008) Data mining based social network analysis from online behavior. In: Tutorial at the 8th SIAM international conference on data mining (SDM' 08)
36. Sun J, Zhu X, Fang Y (2010) A privacy-preserving scheme for online social networks with efficient revocation. In: Proceedings of the 29th conference on information communications, INFOCOM'10. IEEE Press, Piscataway, NJ, USA, pp 2516–2524
37. Tootoonchian A, Saroiu S, Ganjali Y, Wolman A (2009) Lockr: better privacy for social networks. In: Proceedings of the 5th international conference on emerging networking experiments and technologies, CoNEXT '09. ACM, New York, NY, USA, pp 169–180
38. Wasserman S, Faust K (1994) Social network analysis: methods and applications. Cambridge University Press, Cambridge
39. Xie X (2010) Potential friend recommendation in online social network. In: IEEE/ACM international conference on cyber, physical and social computing (CPSCom), pp 831–835, Dec 2010
40. Yang Y, Lutes J, Li F, Luo B, Liu P (2012) Stalking online: on user privacy in social networks. In: Proceedings of the second ACM conference on data and application security and privacy (CODASPY '12), ACM, pp 37–48
41. Yao AC (1982) Protocols for secure computations. In: Proceedings of the 23rd annual symposium on foundations of computer science, IEEE Computer Society, pp 160–164
42. Yao AC (1986) How to generate and exchange secrets. In: Proceedings of the 27th symposium on foundations of computer science, IEEE Computer Society, pp 162–167

Context Based Semantic Relations in Tweets

Ozer Ozdikis, Pinar Senkul and Halit Oguztuzun

Abstract Twitter, a popular social networking platform, provides a medium for people to share information and opinions with their followers. In such a medium, a flash event finds an immediate response. However, one concept may be expressed in many different ways. Because of users' different writing conventions, acronym usages, language differences, and spelling mistakes, there may be variations in the content of postings even if they are about the same event. Analyzing semantic relationships and detecting these variations have several use cases, such as event detection, and making recommendations to users while they are posting tweets. In this work, we apply semantic relationship analysis methods based on term co-occurrences in tweets, and evaluate their effect on detection of daily events from Twitter. The results indicate higher accuracy in clustering, earlier event detection and more refined event clusters.

1 Introduction

Social networking platforms, especially micro-blogging sites such as Twitter, act as an important medium where people share their opinions with their followers. With its millions of users around the world and half a billion tweets per day,[1] textual content in Twitter is an abundant and still growing mine of data for information retrieval

[1] http://news.cnet.com/8301-1023_3-57541566-93/report-twitter-hits-half-a-billion-tweets-a-day

O. Ozdikis (✉) · P. Senkul · H. Oguztuzun
Middle East Technical University, Ankara, Turkey
e-mail: ozer.ozdikis@ceng.metu.edu.tr

P. Senkul
e-mail: pinar.senkul@ceng.metu.edu.tr

H. Oguztuzun
e-mail: oguztuzun@ceng.metu.edu.tr

F. Can et al. (eds.), *State of the Art Applications of Social Network Analysis*, Lecture Notes in Social Networks, DOI: 10.1007/978-3-319-05912-9_2,

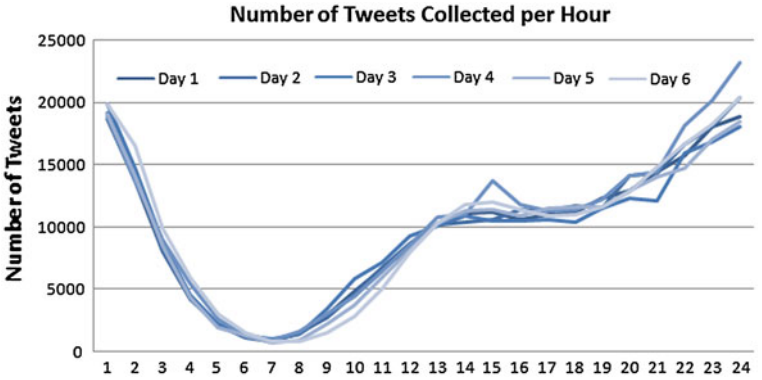


Fig. 1 Number of Tweets per hour collected by using the Twitter streaming API

researchers. This retrieved information is used for analyzing public trends, detecting important events, tracking public opinion or making on-target recommendations and advertisements. However, there may be variations in the contents of such a free and unsupervised platform, even if people refer to the same topic or the same concept in their tweets. Users may express the same thing in their own personal ways, such as by using unusual acronyms or symbols. Geographic, cultural and language diversities cause variations in the content. Even the regional setting or character set used on the device for posting affects uniformity. Moreover, the character limitation of a tweet in Twitter forces people to write in a compact form, possibly with abbreviations or symbols. Last but not least, spelling mistakes are definitely another major reason for divergence in content. For these reasons, in order to apply information retrieval algorithms more effectively in such an environment, it is necessary to be able to figure out the semantic relationships among the postings under the variations. Our goal is to identify such cases, and understand what the user could have meant, so that we can enrich a given tweet with possible similar terms.

In this work, we devise methods to extract semantic relationships among terms in tweets and use them to enhance the event detection capability. For the extraction of semantic associations, we use co-occurrence based statistical methods. Although the techniques we use are independent of language, we limit our scope to Turkish tweets posted in Turkey. In Fig. 1, we present a chart that displays the number of tweets per hour with Turkish content that we collect for several days. This figure indicates that, the collected tweets show a daily pattern and there are almost no postings around 7 in the morning. Therefore, we perform semantic relationship analysis and event detection on daily basis, such that we consider 7 am as the end of the day and identify hot topics of the previous day offline.

Our intuition is that implicit similarities among terms are time-dependent. In other words, consistently with the tweet graph in Fig. 1, we observe that a new day mostly starts with new events and new terms in Twitter. Therefore we choose to analyse term relations, i.e., co-occurrences, within the scope of per day. Using such a context

based relation extraction and applying these relations for tweet expansion, we aim to obtain earlier event detection, with longer lifetime and accurately clustered tweets. Moreover we obtain more refined results so that the users can follow the daily reports more easily. The rest of this chapter is organized as follows:

- We first present several metrics for measuring the associations among terms in a collection of documents (Sect. 2).
- We present our algorithm for event detection and introduce our proposed methods for enhanced event detection (Sect. 3).
- We provide analysis results by evaluating proposed techniques for discovering term associations and enhanced event detection. We discuss the results of our experiments (Sect. 4).
- We present a review of the related work about event detection, especially on social media. Recent studies using semantics in word co-occurrences and use of similarity analysis in different problem domains are given (Sect. 5).
- We finally conclude the chapter with an overview of the results obtained and future work directions (Sect. 6).

2 Term Relationship Metrics

In natural languages, terms can have several types of semantic and grammatical relationships in sentences. For example, a term can be the synonym, antonym, or hyponym of another term. Two closely related terms, such as the first and last names of a person, may occur in sentences more frequently than other term pairs. There are also phrases that are composed of multiple terms frequently used in the same pattern, such as "*take advantage of*", "*make use of*" or "*keep an eye on*". It is possible to look for these relationships in dictionaries, thesauri or encyclopedia. On the Internet, there are even online resources for this purpose such as WordNet and Wikipedia, which are already utilized in studies on similarity analysis and word sense disambiguation [2, 16]. However, in addition to the fact that these online resources are not mature enough for all languages yet, the language of Twitter is remarkably different than the one in dictionaries or newspaper texts. First of all, there is no authority to check the correctness of the content in Twitter. It is a social networking platform where people can write whatever they want in their own way. In its simplest terms, they can make spelling mistakes or they may type a word in different ways (like typing *u* for *ü* or *o* for *ö* in several languages). Therefore, instead of utilizing an online dictionary, we adopt a statistics-based technique to identify associations among a given set of terms. The idea is that semantic relations of terms have some impact on their distribution in a given document corpus. By analyzing syntactic properties of documents, i.e., tweets in this case, associations between term pairs can be extracted. There are several relationship metrics depending on which statistical patterns in term distributions to look for. First order relations, also known as syntagmatic relations, are used to identify term pairs that frequently co-occur with each other [19, 25].

A person's first and last names, or place names such as *United States* or *Los Angeles* can be considered to have this kind of relationship. Moreover, term pairs like *read-book*, *blue-sky*, and *happy-birthday* are examples of first order associations.

Term co-occurrences can be used in order to identify second order relationships as well. Second order associations are referred to as paradigmatic relations and they aim to extract term pairs that can be used interchangeably in documents [19]. If two terms co-occur with the same set of other terms, this can be interpreted as one can be replaced with the other (possibly changing the meaning of the sentence, but this is immaterial). Therefore, methods that find paradigmatic relations do not directly use co-occurrence counts between two terms, but consider the mutuality of co-occurrences with other words. For example, *photo-photograph* or *black-white* are such word pairs that most probably co-occur with the same words.

In addition to the first and second order associations, there are also higher order associations with basically the same logic [5]. For example, if there is a high number of co-occurrences among the term pairs $t_1 - t_2$, $t_2 - t_3$ and $t_3 - t_4$, then t_1 and t_4 can be considered as having a third-order association. In this work, we focus on first and second order relationships. Finding these relationships can be achieved by using several metrics. For syntagmatic relations, a straightforward measurement is simply counting the co-occurrences of term pairs. Other co-occurrence ranking metrics are proposed such as Dice, Cosine, Tanimoto and Mutual Information [10]. Application of entropy-based transformations [20] or Singular Value Decomposition [7] on the co-occurrence frequencies are also further improvements for first-order relations. For finding second order relationships between two terms t_1 and t_2, one of the metrics is to count the number of distinct terms t_3 that co-occur with t_1 and t_2 [5]. However, in our work, we apply a method based on the comparison of co-occurrence vectors as presented in [19]. The basic idea is to generate term co-occurrence vectors and compare their similarity in the vector space. We experiment with cosine and city-block distances for similarity comparisons. For first order relations, we simply count the number of co-occurrences, i.e., raw co-occurrence values. Both in the first and second order association analysis, our objective is not only finding the most related terms, but also assigning them a similarity score, i.e., a value between 0 and 1. This similarity score will be used while applying a lexico-semantic expansion to tweet vectors, as will be explained shortly.

Statistical methods do not necessarily require a dictionary or human annotated data. First of all, this brings about language independence. Moreover, by analyzing terms in specific timeframes, ambiguities can be resolved depending on the context. For example, the term *goal* could have many related terms, such as *match*, *objective* or *end*. But if we know that there was an important soccer game that day and the term appears very frequently in tweets, then we can assume that it is used in sports context, therefore *match* would be the most suitable related term.

Regarding the performance issues and resource limitations, we do not apply semantic analysis on rare terms. On a daily collection of around 225 K tweets, there are more than 140 K distinct words on average, some of them appearing only in a few tweets. Moreover, in order to capture co-occurrences, content-rich tweets are preferred. Therefore, we process tweets with at least 4 terms and compare the terms with

minimum frequency of 300. These numbers are selected by intuition, after observing the Twitter traffic for a period of time. They can be adapted for another language, another tweet selection criterion, or another processing infrastructure. Here we would like to emphasize that our focus in this work is to demonstrate that the extraction of semantic relations in an uncontrolled environment such as Twitter is practical for better event detection. Finding the most suitable parameter values or most efficient similarity metrics could be the objective of another study.

2.1 First Order Relationships

As explained before, in order to find the first order relationships, we use the raw co-occurrence values. In our previous work [13], after finding the number of times the term pairs co-occur, we identified the semantically related term pairs if they appear in more than 50 tweets together. Moreover, if two terms are found to be semantically related, we assigned a constant similarity score of 0.5. In this work, we developed a more generic solution and adopted the approach that we used for discovering hashtag similarities in [14]. Instead of using a threshold for deciding the similarity of two terms and giving them a constant similarity score, we assign normalized similarity scores for each term pair by using their co-occurrence values. For example, the term pair with the maximum number of co-occurrence value, c_{max}, on a given day has the similarity score 1.0. For other term pairs t_i and t_j with a co-occurrence count of $c_{i,j}$, their similarity score is given by the ratio of $c_{i,j}/c_{max}$.

2.2 Second Order Relationships with Cosine Similarity

For the second order relationships, term co-occurrence vectors are generated. Let $c_{i,j}$ represent the number of co-occurrences of the terms t_i and t_j. Then, for each term t_i, we count its co-occurrences with other terms t_1, t_2,... $t_{|w|}$ where W is the set of distinct terms collected on that day's tweets. After forming the term vectors as given in (1),

$$\mathbf{t_i} = (c_{i,1}, c_{i,2}, ..., c_{i,i-1}, 0, c_{i,i+1}, ..., c_{i,|W|-1}, c_{i,|W|}) \tag{1}$$

we compare their cosine distance by using the cosine distance equation in (2) [28].

$$sim_{\text{cosine}}(\mathbf{t_i}, \mathbf{t_j}) = \frac{\mathbf{t_i} \cdot \mathbf{t_j}}{|\mathbf{t_i}||\mathbf{t_j}|} = \frac{\sum_{k=1}^{|W|} c_{i,k} c_{j,k}}{\sqrt{\sum_{k=1}^{|W|} c_{i,k}^2 \sum_{k=1}^{|W|} c_{j,k}^2}} \tag{2}$$

Again we do not use any threshold for the decision of similarity but rather use the cosine distance as the similarity score, which is already in the range [0,1].

2.3 Second Order Relationships with City-Block Distance

City-block distance is another simple vector comparison metric [19]. After forming the co-occurrence vectors, while comparing two vectors, it finds the sum of absolute differences for each dimension as given in (3).

$$sim_{city\text{-}block}(\mathbf{t_i}, \mathbf{t_j}) = \sum_{k=1}^{|W|} |c_{i,k} - c_{j,k}| \tag{3}$$

Similar to the solution we applied for first order relations, we normalize the distances in [0, 1] and use these values as similarity scores.

3 Event Detection and Semantic Expansion

In this work, we perform offline event detection on tweets. However the algorithms we implement can also be used online with further performance optimizations. The flow of the event detection process is depicted in Fig. 2. Dashed arrows indicate the extension that we implemented on a traditional clustering algorithm. We first present the data collection, tweet vector generation, clustering and event detection steps. Then we explain how we carry out lexico-semantic expansion and improve event detection quality.

For tweet collection from the Twitter Streaming API, we use Twitter4J,[2] a Java library that facilitates the usage of Twitter API. We apply a location filter and gather tweets posted by users in Turkey, with Turkish characters. Posts with other character sets such as Greek or Arabic letters are filtered out. The gathered tweets are immediately stemmed with a Turkish morphological analyzer called TRMorph [6]. After further preprocessing, including the removal of stop words and URLs, they are stored into the database. Using this process, we collect around 225 K tweets per day. Further details regarding the tweet collection and preprocessing steps are found in our previous work [13].

Our event detection method is an implementation of agglomerative clustering, applied on tweets collected in a given period of time. In this algorithm, tweets are represented by tweet vectors that are generated by the TF-IDF values of the terms in each tweet. In order to fix the size of these vectors and calculate the IDF values of terms, all tweets in the given period are pre-processed. The number of distinct terms, i.e., the dimension of tweet vectors, is determined and document frequencies of each term are found. Finally tweet vectors are created by using the frequencies of their terms and their inverse document frequencies.

The clustering algorithm simply groups similar tweets according to the distance of their tweet vectors in n-dimensional vector space. Just like tweet vectors, clusters

[2] Twitter4J homepage, http://twitter4j.org.

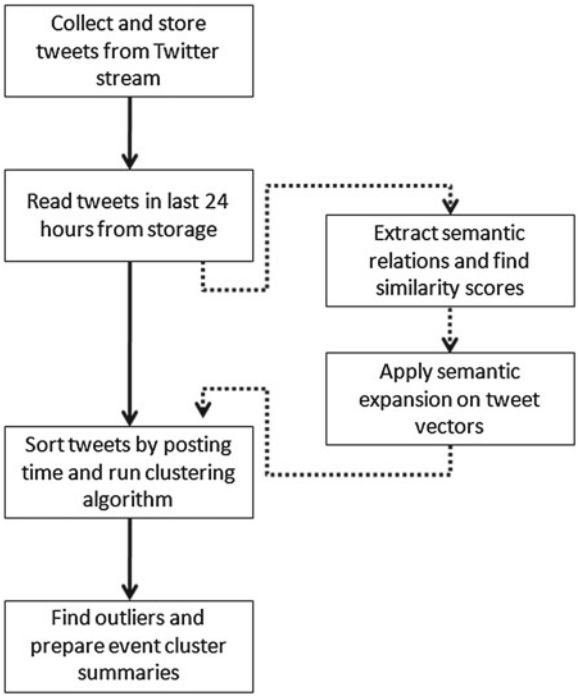


Fig. 2 Event detection process

are also represented with vectors. A cluster vector is the arithmetic mean of the tweet vectors grouped in that cluster. Tweets are processed one by one according to their posting time. For each tweet, the most similar cluster vector is found. For similarity calculation, we use cosine distance as given in Eq. (2). If the similarity of the most similar cluster is above a given threshold, the tweet is added to that cluster and the cluster vector is updated. Otherwise, that tweet starts a new cluster on its own. If no tweet gets added to a cluster for a certain period of time, then it is finalized, meaning that the event does no longer continue.

Finally, with all tweets processed, we apply an outlier analysis using the empirical rule (also known as the three-sigma or 68-95-99.7 rule) [14, 29]. According to this rule, we first find the mean number of tweets in clusters and their standard deviation (σ). We mark the clusters with more than mean+3σ tweets as *event clusters*. In order to present an understandable summary to the user, summaries are generated for each event cluster. Summaries are simply the first three terms in cluster vectors with the highest TF-IDF values.

The event detection method we introduced above is our basis method, whose path is indicated with the solid arrows in Fig. 2. We refer to it as BA (Basis Algorithm) in the rest of the chapter. The novelty of our work is an extension on this basis algorithm by applying a lexico-semantic expansion to tweets. This extension is composed of two

steps, namely calculation of similarity scores among frequent terms and using these scores while applying the semantic expansion on tweet vectors before feeding them to the clustering process. As explained before, we implemented three metrics for the analysis of term associations. We evaluate their results and apply them on clustering separately. We label these clustering implementations as FO for First Order, COS for Cosine and CBL for City-Block metrics.

The first step in this extension is the calculation of similarity scores among term pairs that appear in the tweets to be clustered. By using one of the abovementioned metrics, we obtain term–term similarity scores. For each term, we keep only top-n similarities, due to the performance optimizations. We choose to use top-3 similarities in our experiments.

After having the similarity scores, generated either with first order or second order analysis, we use these similarity scores to apply semantic expansion on tweet vectors. The idea of semantic expansion is similar to the studies in [9] and [16]. In this work, we develop specialized methods for evaluating the numerical values. The expansion process is as follows:

1. Given a tweet t^i and its tweet vector with k terms $[t^i_1, t^i_2, ...t^i_k]$ with corresponding TF-IDF weights $[w^i_1, w^i_2, ...w^i_k]$,
2. For each t^i_x in the tweet vector,
    a. Search for its semantically related terms. Let t^i_x be associated with the three terms t_a, t_b, t_c with similarity scores s_a, s_b, and s_c
    b. Find the products $w^i_x s_a$, $w^i_x s_b$, and $w^i_x s_c$ as expanded TF-IDF values
    c. If t_a does not exist in the tweet vector or if its TF-IDF value in the tweet vector is less than the expanded TF-IDF value, namely $w^i_x s_a$, then insert t_a to the tweet vector with its expanded TF-IDF value. Otherwise, simply ignore it. That means if a term already exists in the tweet with a high TF-IDF value, then it is not changed by the semantic expansion process. Do this step for t_b and t_c as well.

Such an expansion usually results in tweet vectors with much higher dimensions than the original ones. The effect of this may be larger clusters with more similar tweets, or larger clusters with junk tweets. Therefore it is important to identify correct term similarities with correct similarity scores. As elaborated in the following section, application of such a semantic expansion has several advantages.

4 Evaluation and Results

Our evaluations focus on two core functions implemented in our enhanced event detection method, namely the success of identified term similarities and the quality of detected events. Our 3-week test dataset is composed of about five million tweets posted between the 4th and 25th of September 2012 with Turkish content in

Turkey. We adopt daily processing of tweets, i.e., tweets collected during one day are processed at 7am in the following morning. The results are evaluated in accordance with this regular daily processing, presented for each day covered by our test dataset.

4.1 Evaluation of Term Similarity Analysis

Before presenting the similarity analysis, we present some figures to clarify the statistical value of our evaluation. The average number of tweets per day, collected during the generation of our test dataset is 225,060. On average, 140 K distinct terms are used in Twitter every day, with 518 of them deemed as having high frequency by our definition (>300). It means that we compare about 518 terms on average every day for their semantic associations.

For evaluating the success of a semantic relationship extraction method, it is possible to make use of word similarity questions in a language exam such as TOEFL [19, 20]. Comparing the results of the algorithm with the answer sheet of the exam is an acceptable method. However, terms written in Twitter are hardly found in a dictionary due to the variances in writing conventions and spelling mistakes. Moreover, ambiguous similarities can exist depending on the context, as we have explained in an example where the term *goal* must be related to the term *match* if there is a soccer game that day. Discovery of such context-based associations may not always be captured by an automatic language test.

In order to set a golden standard for evaluation, we prepared a questionnaire per day that is composed of 120 questions for each day (2,520 questions in total). They were five-choice questions, where for a given term, users were asked to mark the most relevant term among the choices. Choices for each question are populated from the results of three similarity measurement algorithms (the ones with highest similarity scores) and two randomly selected terms from the corpus. If similarity estimations of two algorithms coincide, we write that term only once in the choices, and another random term is inserted to present five distinct choices for the question. Questions are answered by seven users who are native Turkish speakers and mostly computer science graduates. We explained them briefly our relevance criteria, such as spelling mistakes (*günaydın* $\sim$ *gunaydin*), abbreviations (*Barca* $\sim$ *Barcelona*), and strong association of domains (*sleep* $\sim$ *dream*). Users could also select "none", if they think there is no related term among the choices or if the term in the question may not have a matching term (e.g. it can be a number, a symbol, or some meaningless word which makes it impossible to decide similarity). Therefore, unanswered questions could mean that either it is inapplicable, or none of the similarity algorithms could successfully find a similar term. We compare the similarity algorithms only on the answered questions.

While grading the success rates of the algorithms, we count the number of correctly estimated similar terms. If the term with the highest similarity score found by an algorithm is the same as the term marked by the user, it is accepted as a correctly answered question by that algorithm. There may be cases where all three algorithms

Table 1 Offline term similarity results

Day	Marked questions	Accuracy ratio		
		FO (%)	COS (%)	CBL (%)
1	63	**62**	25	25
2	53	**38**	36	30
3	73	**44**	36	34
4	66	35	**50**	15
5	68	**44**	28	21
6	68	**49**	32	25
7	73	**52**	27	29
8	84	**63**	19	29
9	77	**38**	36	27
10	62	**61**	31	21
11	58	**48**	41	21
12	67	**49**	30	6
13	71	**56**	34	14
14	69	**54**	33	23
15	70	**54**	29	24
16	90	**47**	33	17
17	82	**41**	30	22
18	101	**42**	30	17
19	97	**48**	31	16
20	65	**43**	32	18
21	58	**48**	31	16
Avg.	72	**48**	32	21

find the same correct result for a question. Then they all get their points for that question. As a result, the ratio of the number of correct results of each algorithm to the number of marked questions is used to evaluate the accuracy of the algorithms. The results of term similarity evaluations are presented in Table 1. The column labeled with "Marked questions" indicates the number of questions with a valid answer, i.e., the user did manage to select an answer among the choices. The percentages of correct results for each algorithm are presented in the rest of the columns, where the highest accuracy ratio for each question is highlighted. Results are presented for each day in the test data set.

The number of answered questions shows that, users could find a relevant term among the choices for more than half of the questions. Ignoring the luckily selected random terms while generating the choices of the questions, this can be interpreted as at least one algorithm finds a relevant term in the corpus at least half of the time. According to this table, the first order relations are apparently closer to the users' answers. Several examples of successfully detected first order relations are *birth* $\sim$ *day*, *listen* $\sim$ *music*, and *read* $\sim$ *book*. Obviously, given the first term, these can be considered as the first mappings that come to mind even without any multiple

choices. In other words, they are easier to identify. Second order relations are a little harder for a person to see at first. Several examples of the correctly selected second order term pairs are *morning* ∼ *early*, *happy* ∼ *fine*, and *class* ∼ *school*. Therefore we believe the accuracy ratio of the second order associations should not be underestimated. Between cosine distance and city-block distance, cosine similarity makes more accurate guesses.

Although this table can give an idea about the power of co-occurrence based similarity techniques, calculated similarity scores play an important role for the event detection and expansion algorithm. The effect of similarity scores is better observed in the evaluation of event detection.

4.2 Evaluation of Event Detection

In the Topic Detection and Tracking (TDT) domain, evaluation of event detection algorithms is usually made either by precision-recall [23] or by false alarm-miss rate analysis [3, 8]. In our work, we use the precision-recall metric for evaluation. Moreover, we analyzed the event detection times and studied the refinement of the generated event clusters. The comparison of the event detection algorithms mainly focuses on the following three goals:

1. Number of event clusters and their tweets: In order to present a useful and readable report to the users, it is preferable to generate more refined event clusters with as many correctly clustered tweets as possible. The best case would be presenting unique event clusters for each actual event in the real world, with explanatory cluster summaries.
2. Accuracy of event clusters: Ideally, there should be no irrelevant tweet in an event cluster, and an event cluster should cover as many relevant tweets as possible. Therefore, our goal is to improve the accuracy of tweets in event clusters.
3. Time span of detected events: Our expectation is that the utilization of hidden semantic relations among terms should result in earlier generation of event clusters with longer duration. Especially longer duration of larger clusters shift their finalization to a later time. Therefore, they can attract unique tweets that would normally start new clusters on their own.

Our event detection evaluation is based on human annotated tweets. For each day in our test data set, we run four event detection algorithms, where our baseline is the one with no semantic expansion (BA). The results are compared with the other algorithms using different expansion methods, namely first order (FO), second order with cosine (COS) and city-block distances (CBL). An event detection algorithm may find several event clusters about an event. We consider them all as belonging to the same event, which means that an event may be represented by one or more event clusters in an algorithm. While matching an event cluster with an event, we consider its cluster summary. We would like to remind that the cluster summaries are in fact

the top three terms in the event cluster with the highest TF-IDF values. In order to assign an event cluster to an event, its summary must be understandable and clearly mention the corresponding event.

After the event clusters are obtained, we select one event per day as our "target event" for that day. While determining a target event cluster, we utilized other resources in Internet such as TV rating results or newspapers in order to determine the most popular and important event for each day. On the other hand, we observed that people are posting tweets that are not newsworthy, such as "good morning" or "good night" tweets. It is possible to apply a classification algorithm in order to filter out such event clusters [4, 22]. We simply let them survive as event clusters but do not consider them as target events in our evaluations.

Although our event detection algorithm is executed on a daily basis at 7 am, events do not have to last one day. According to our observations, almost no event lasts longer than two days though. In fact, 2-day events are observed only when the event happens at midnight. Since we process tweets at 7am, an event that happened around midnight is usually detected on both days. For example, the first target event in our test data set is the tragic loss of a young soccer player, Ediz Bahtiyaroglu, who had played in several major Turkish soccer clubs and passed away at the age of 26 from heart attack. As soon as this event is heard at night, people posted tweets to express their sorrow and condolences. These tweets continued during the day, which resulted in detection of this event the next day as well. Apart from this, other target events that we marked for evaluation can be classified as soccer games, anniversaries of historical events, disasters, criminal incidents, popular TV shows or news about celebrities, which are mentioned by thousands of people in Turkey.

We first present the number of event clusters, number of annotated tweets, and the results of our precision-recall analysis in Table 2 for each day in our test data. The table can be interpreted as follows. The first column labeled as "Day" displays the index of the day between 4th and 25th of September 2012 (e.g. Day-1 is the 4th of September). The row "avg" is the average of all days for the feature in the corresponding column. The second column, namely "Annt.", represents the number of tweets that are manually annotated by human annotators as belonging to the target event of that day. By "annotation", we mean manually marking a tweet to be related to an event or not. We adopt the following method to generate a tweet set to annotate. Given a day, each algorithm that we implement generates their event clusters for the target event for that day. Assume the set of tweets grouped by each of these algorithms are T_{BA}, T_{FO}, T_{COS}, and T_{CBL}. Then the tweets that we manually annotate are the union of these tweet sets, i.e., $T_{BA} \cup T_{FO} \cup T_{COS} \cup T_{CBL}$. Consider Day-20 for example. The target event of that day was the game of a popular sports club in Turkey. Each algorithm detected multiple event clusters for that target event, and the number of tweets clustered in these event clusters were 574, 427, 674, and 519 respectively for each algorithm. As shown in the table, the union of these tweet sets is composed of 862 tweets, which gives the tweets to be annotated for that day's target event.

The third column group that is labeled as "Target Event Cluster Ratio" displays the number of target event clusters and the number of event clusters detected for that

Table 2 Offline event detection results

Day	Annt.	Target event cluster ratio				Precision				Recall				F-Score			
		BA	FO	COS	CBL	BA	FO	COS	CBL	BA	FO	COS	CBL	BA	FO	COS	CBL
1	495	2/15	2/13	2/11	2/11	**1.00**	0.99	0.98	0.98	0.50	0.70	0.72	**0.78**	0.66	0.82	0.83	**0.87**
2	2356	2/12	2/4	2/9	3/7	**0.98**	0.48	0.93	0.94	0.30	**0.77**	0.68	**0.77**	0.46	0.59	0.79	**0.85**
3	3907	7/14	7/13	6/13	5/8	**0.97**	0.94	0.88	0.85	0.50	0.63	**0.79**	0.61	0.66	0.76	**0.83**	0.71
4	1314	4/21	5/17	2/16	2/14	0.99	0.88	**1.00**	0.91	0.45	**0.79**	0.26	0.32	0.62	**0.83**	0.41	0.48
5	561	1/18	1/7	1/11	1/12	**1.00**	0.66	0.74	0.79	0.24	0.71	0.71	**0.72**	0.39	0.69	0.72	**0.75**
6	2367	1/6	2/5	2/6	3/9	**1.00**	0.68	0.91	0.88	0.57	**0.92**	0.81	0.78	0.72	0.78	**0.86**	0.83
7	769	2/11	2/12	1/11	1/8	**0.94**	0.92	0.92	0.83	0.42	0.43	0.72	**0.88**	0.58	0.58	0.81	**0.86**
8	4014	7/15	7/12	8/12	11/16	0.59	0.54	**0.62**	0.59	0.51	0.58	**0.72**	0.66	0.55	0.56	**0.67**	0.62
9	288	1/16	1/13	1/15	2/16	**1.00**	0.95	**1.00**	0.78	0.57	0.62	0.57	**0.75**	0.73	**0.76**	0.72	**0.76**
10	800	1/14	1/14	2/11	1/10	**1.00**	**1.00**	0.92	0.92	0.53	0.51	0.49	**0.90**	0.69	0.67	0.64	**0.91**
11	869	2/14	0/4	3/17	3/15	0.64	NaN	0.65	**0.67**	0.38	NaN	0.74	**0.94**	0.48	NaN	0.69	**0.78**
12	1103	1/16	2/11	1/6	2/5	**1.00**	0.76	0.65	0.61	0.33	0.53	0.73	**0.81**	0.50	0.63	0.59	**0.70**
13	1404	2/9	2/8	2/9	5/12	**1.00**	**1.00**	0.97	0.98	0.66	0.66	**0.70**	0.65	0.80	0.80	**0.82**	0.78
14	822	2/10	1/6	1/11	2/6	0.46	**0.98**	**0.98**	0.44	0.45	0.63	0.60	**0.85**	0.46	**0.76**	0.74	0.58
15	1233	4/16	1/9	1/10	2/9	0.98	**1.00**	0.99	0.97	0.36	0.15	0.41	**0.72**	0.53	0.26	0.58	**0.83**
16	4550	5/11	2/4	5/9	5/8	**0.99**	0.84	0.96	0.90	0.47	0.58	**0.63**	0.51	0.64	0.69	**0.76**	0.65
17	4110	5/10	5/7	4/5	5/8	**0.99**	0.94	0.86	0.89	0.63	**0.76**	0.63	0.57	0.77	**0.84**	0.73	0.69
18	313	2/19	1/11	1/12	0/10	**1.00**	**1.00**	**1.00**	NaN	**0.91**	0.48	0.35	NaN	**0.95**	0.65	0.52	NaN
19	326	1/16	1/13	2/12	2/11	**0.99**	0.61	0.71	0.77	0.63	0.67	**0.93**	0.88	0.77	0.64	**0.81**	0.82
20	862	4/15	2/9	3/12	5/13	**1.00**	0.87	0.79	**1.00**	**0.81**	0.52	0.75	0.73	**0.89**	0.65	0.77	0.84
21	251	2/10	0/9	1/12	0/5	0.40	NaN	**0.99**	NaN	**0.92**	NaN	0.64	NaN	0.56	NaN	**0.77**	NaN
avg	1557	2.8/13.7	2.2/9.6	2.4/11.0	2.9/10	**0.90**	0.84	0.88	0.83	0.53	0.61	0.65	**0.73**	0.64	0.68	0.72	**0.75**

day. For example, on Day-20, 15 event clusters were identified as outliers by the BA algorithm. Among these 15 event clusters, we found four of them to be related with the target event, namely the soccer game. These numbers give an idea about the information contained in event clusters if considered together with the coverage. The lowest number of event clusters with the highest coverage ratio leads to more understandable event cluster summaries for people. Otherwise, if too many event clusters are generated with low coverage, this would mean scattered information with poor comprehensibility.

Rest of the columns in the table present the precision, recall, and F-score values for each day for all clustering algorithms that we implement. If an algorithm finds no event cluster for a target event, then its precision-recall analysis becomes inapplicable, denoted as NaN. This accuracy analysis can be interpreted as follows: The basis algorithm usually results in better precision, 0.90 on average. This is because we apply no semantic expansion to the tweets, and there is less chance for an irrelevant tweet to be included in a cluster. On the other hand, its coverage is not as large as the algorithms using semantic expansion. Whether it is the first or second order relationship, semantic expansion techniques usually cover more tweets than the basis algorithm. The overall accuracy is calculated by using the F-score equation given in (4). According to these results, second order associations provide higher accuracies.

$$Fscore = \frac{2 \times precision \times recall}{precision + recall} \tag{4}$$

In addition to these analyses, another criterion for better event detection is the time of detection. It is preferable to hear about an event as soon as possible. As an example, we present the active times of event clusters of Day-20 in Fig. 3. An active event cluster is depicted in tiles, with its beginning and ending times corresponding to the earliest and latest tweet times, respectively. The time window covered in each algorithm is highlighted in gray. The first group of rows is the target event clusters found by the BA algorithm. As given in Fig. 3, the BA algorithm detected four event clusters for the soccer game that day. Therefore the figure displays four lines for BA with the clusters' active time windows. According to this algorithm, the first tweet about the game was posted at around 14:00. In other words, the track of this event was first observed in the afternoon and lasted for about 2 h. Then, no similar tweet has been observed until the evening. It can also be said, the content of tweets after 16:00 were not very related to the first event cluster. Then at about 20:30, the first goal was scored in the game, which cause the generation of two event clusters at that time. The name of the scoring player was *Burak Yılmaz*, which was also spelled as *Burak Yilmaz* in some tweets (the letter *i* replacing *ı*). We consider this as the most important reason for two event clusters about the same event. The last event cluster generated by BA begins at 10 pm and lasts for three hours. The summary of this event cluster is about the celebrations after the end of the match.

In the first three algorithms, the event detection times are almost the same. On the other hand, the duration of the first event cluster is longer for FO and COS algorithms. This is considered to be the result of successfully identified semantic associations.

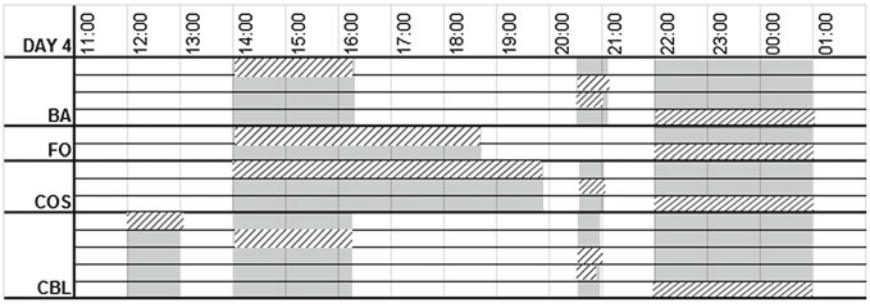


Fig. 3 Timespan of event clusters on day-20

Interestingly in the COS algorithm, there is only one event cluster detected at 20:30. Apparently the algorithm found a high similarity score between the terms *Yılmaz* and *Yilmaz*, which results in a single event cluster at that time. This observation highlights one of the major objectives of our research, i.e., elimination of duplicate event clusters. Another objective was earlier detection of events. In this specific case on Day-20, this is achieved by using the CBL algorithm. The first event cluster generated for this event starts at about 12:00, which is 2 h earlier than the earliest event detection times of other algorithms.

5 Related Work

Event Detection, also known as Event Detection and Tracking, aims to identify unique events by processing textual materials such as newspapers, blogs, and recently, the social media [1]. Especially after the foundation of Twitter in 2006 and with its millions of users around the world today, there have been many studies that utilize peoples' postings for information retrieval purposes [11, 12]. An implementation of real-time event detection from Twitter is described in [22]. In that work, Sankaranarayanan and coworkers follow tweets of handpicked users from different parts of the world; cluster them for event detection, and assign a geographic location to the event in order to display it on a map. In a similar study, Sakaki and coworkers focus on earthquakes in Japan [21]. They detect earthquakes and make estimations about their locations only by following tweets in Twitter. There are also studies for improving first story detection algorithms on Twitter, i.e., identifying the first tweet of a new event [17]. An interesting use of event detection in Twitter is presented in [15]. In that work, Park and coworkers aim to detect important events related to a baseball game, and display annotated information to people that watch that game on TV. This can be considered to be a similar case of our example in Fig. 3. By detecting separate events for scoring a goal, or making a homerun as stated in [15], it is possible to retrieve who made the homerun at what time and where.

Using the semantics in word co-occurrences has been exploited in several studies on event detection. In [23], authors implement a solution by integrating burst detection and co-occurrence methods. In that solution, they track a list of terms (entities which may be taken from a reference work like Wikipedia) in query logs or tweet data in order to detect extraordinary increases in their appearances. They argue that if two entities show unusually high frequencies (bursts) in the same time-window, they possibly belong to the same event. In order to measure their relevance and group in the same event, content-rich news articles in the corresponding time window are processed and first order associations among terms are analyzed. A similar approach is used in [24]. The intuition is that documents about an event should contain similar terms. By generating a graph with terms as nodes, and co-occurrences as edges, they identify highly connected sub-graphs as event clusters.

Apart from event detection purposes, similarity analysis on textual materials can be useful for recommendation and applying intelligent expansion to queries. In [27], authors study the spatio-temporal components of tweets and identify associations among trending topics provided by Twitter API. They generate vectors considering their spatial and temporal aspects, which are then pairwise compared with Euclidean distance to find the most similar topic pairs in Twitter. In a more recent work, methods in [27] are extended with the similarities of topic burst patterns in order to take event-based relationships into account [26]. The intuition is that two similar topics should have similar bursting periods as well as spatial and temporal frequency similarities.

A method for exploring associations among hashtags in Twitter is presented in [18]. The proposed model aims to make more complex temporal search requests in Twitter, such as asking for hashtags with an increasing co-occurrence with a given hashtag in a desired period of time. Another example of analyzing term associations is given in [10], where users are guided while giving a title for the product they want to sell in an online store. Text entered by a seller is compared with previous queries of buyers, and a better title for the product is recommended to the seller.

In our preliminary work, we presented the basics of the techniques that we elaborate in this chapter, executed on a very limited data set of three days [13, 14]. These techniques have been combined and extended for context-based daily event detection, tested on a much larger data set annotated by several users. Moreover, detailed evaluations focus on both term similarity analysis and event detection methods, providing a more reliable and clear overview of results.

6 Conclusion

In this work, we aim to extract associations among terms in Twitter by using their co-occurrences and use them in a semantic expansion process on tweets in order to detect events with higher accuracy, with larger time span, and in a user-friendly form. We improve our previous work by using similarity scores instead of thresholds and constant multipliers for semantic expansion. Moreover, we identify context-dependent associations by evaluating terms in specific time windows. Daily event

clusters are determined by making an outlier analysis. Although our methods are applied on tweets in each day, they can be adapted to work in different time granularities or in an online system, which we are planning to implement as a future work. Moreover, we would like to experiment periodically merging and/or dividing event clusters in the course of event detection in order to improve the resulting event clusters.

Our methods are tested on a set of around five million tweets collected in three weeks with Turkish content. We implemented three different semantic similarity metrics and evaluated them on this test data set. Results of these metrics are further analyzed in the evaluations of our event detection methods. Improvements are observed in event detection in several aspects, especially when second order associations are used. As the methods we implement do not require a dictionary or thesaurus, they can be used for other languages as well.

Acknowledgments This work is supported by TUBITAK with grant number 112E275.

References

1. Allan J, Carbonell J, Doddington G, Yamron J, Yang Y (1998) Topic detection and tracking pilot study: final report. In: Proceedings of the DARPA Broadcast News transcription and understanding, Workshop
2. Banerjee S, Pedersen T (2002) An adapted Lesk algorithm for word sense disambiguation using wordNet. In: Lecture notes in computer science, vol 2276, pp 136–145
3. Can F, Kocberber S, Baglioglu O, Kardas S, Ocalan HC, Uyar E (2010) New event detection and topic tracking in Turkish. J American Soc Inf Sci Technol 61(4):802–819
4. Castillo C, Mendoza M, Poblete B (2011) Information credibility on Twitter, In: Proceedings of World Wide Web conference
5. Chakraborti S, Wiratunga N, Lothian R, Watt S (2007) Acquiring word similarities with higher order association mining. In: Lecture notes in computer science, vol 4626
6. Coltekin C (2010) A freely available morphological analyzer for Turkish, In: Proceedings of the 7th international conference on language resources and evaluation (LREC)
7. Deerwester S, Dumais ST, Furnas GW, Landauer TK, Harshman R (1990) Indexing by latent semantic analysis. J American Soc Inf Sci 41(6):391–407
8. Fiscus JG, Doddington GR (2002) Topic detection and tracking evaluation overview. In: Allan J (ed) Topic detection and tracking: event-based information organization. Kluwer Academic Publishers, Dordrecht, pp 17–31
9. Frasincar F, IJntema W, Goossen F, Hogenboom F (2011) A semantic approach for news recommendation, In: Business intelligence applications and the web: models, systems and technologies. IGI Global, Hershey
10. Huang S, Wu X, Bolivar A (2008) The effect of title term suggestion on e-commerce sites. In: Proceedings of the 10th ACM workshop on Web information and data management (WIDM), pp 31–38
11. Java A, Song X, Finin T, Tseng B (2007) Why we Twitter: understanding microblogging usage and communities. In: Proceedings of SNA-KDD. Workshop on Web mining and social network analysis. San Jose, California, pp 56–65
12. Milstein S, Chowdhury A, Hochmuth G, Lorica B, Magoulas R (2008) Twitter and the micro-blogging revolution. OReilly Radar report. http://weigend.com/files/teaching/haas/2009/readings/OReillyTwitterReport200811.pdf

13. Ozdikis O, Senkul P, Oguztuzun H (2012) Semantic expansion of Tweet contents for enhanced event detection in Twitter. In: International conference on advances in social networks analysis and mining (ASONAM), Istanbul, Turkey
14. Ozdikis O, Senkul P, Oguztuzun H (2012) Semantic expansion of hashtags for enhanced event detection in Twitter. Workshop on online social systems (WOSS). Istanbul, Turkey
15. Park H, Youn SB, Lee GY, Ko H (2011) Trendy episode detection at a very short time granularity for intelligent VOD service: a case study of live baseball game, In Proceedings of EuroITV
16. Pembe FC, Say ACC (2004) A Linguistically motivated information retrieval system for Turkish. In: Lecture notes in computer science, vol 3280, pp 741–750
17. Petrovic S, Osborne M, Lavrenko V (2010) Streaming first story detection with application to Twitter. In: Proceedings of the 11th conference of the North American chapter of the association for computational linguistics (NAACL HLT), Los Angeles, California
18. Plachouras V, Stavrakas Y (2012) Querying term associations and their temporal evolution in social data. Workshop on online social systems (WOSS). Istanbul, Turkey
19. Rapp R (2002) The computation of Word associations: comparing syntagmatic paradigmatic approaches. In: Proceedings of COLING, Taiwan
20. Rapp R (2004) A freely available automatically generated thesaurus of related words. In: Proceedings of 4th international conference on language resources and evaluation (LREC), Portugal
21. Sakaki T, Okazaki M, Matsuo Y (2010) Earthquake shakes Twitter users: real-time event detection by social sensors. In: Proceedings of the 19th international conference on World Wide Web, North Carolina, USA
22. Sankaranarayanan J, Samet H, Teitler BE, Lieberman MD, Sperling J (2009) TwitterStand: news in Tweets. In: Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems, Seattle, Washington, 04–06 Nov 2009
23. Sarma AD, Jain A, Yu C (2011) Dynamic relationship and event discovery. In: Proceedings of WSDM '11, pp 207–216
24. Sayyadi H, Hurst M, Maykov A (2009) Event detection and tracking in social streams. In: Proceedings of ICWSM 2009, San Jose CA, USA
25. Schutze H, Pedersen J (1993) A vector model for syntagmatic and paradigmatic relatedness. Making sense of words. In: Proceedings of the conference England, Oxford, pp 104–113
26. Song S, Li Q, Bao H (2012) Detecting dynamic association among Twitter topics. WWW 2012 poster presentation, Lyon, France, 16–20 Apr 2012
27. Song S, Li Q, Zheng N (2010) A spatio-temporal framework for related topic search in microblogging, In: Proceedings of the 6th international conference on active media technology, Toronto, Canada
28. Tan PN, Steinbach M, Kumar V (2006) Introduction to data mining. Addison Wesley, Reading, p 75
29. Wang M, Hsu P, Chuang YC (2011) Mining workflow outlier with a frequency-based algorithm. J Control Autom 4(2)

Fast Exact and Approximate Computation of Betweenness Centrality in Social Networks

Miriam Baglioni, Filippo Geraci, Marco Pellegrini and Ernesto Lastres

Abstract Social networks have demonstrated in the last few years to be a powerful and flexible concept useful to represent and analyze data emerging from social interactions and social activities. The study of these networks can thus provide a deeper understanding of many emergent global phenomena. The amount of data available in the form of social networks is growing by the day. This poses many computational challenging problems for their analysis. In fact many analysis tools suitable to analyze small to medium sized networks are inefficient for large social networks. The computation of the *betweenness centrality* index (BC) is a well established method for network data analysis and it is also important as subroutine in more advanced algorithms, such as the Girvan-Newman method for graph partitioning. In this chapter we present a novel approach for the computation of the *betweenness centrality*, which speeds up considerably Brandes' algorithm (the current state of the art) in the context of social networks. Our approach exploits the natural sparsity of the data to algebraically (and efficiently) determine the betweenness of those nodes forming trees (tree-nodes) in the social network. Moreover, for the residual network, which is often of much smaller size, we modify directly the Brandes' algorithm so that we can remove the nodes already processed and perform the computation of the shortest paths only for the residual nodes. We also give a fast sampling-based algorithm that computes an approximation of the betweenness centrality values of the residual network while returns the exact value for the tree-nodes. This algorithm

M. Baglioni · F. Geraci · M. Pellegrini (✉)
Istituto di Informatica e Telematica del CNR, Via G. Moruzzi 1, 56100 Pisa, Italy
e-mail: m.baglioni@iit.cnr.it

F. Geraci
e-mail: f.geraci@iit.cnr.it

M. Pellegrini
e-mail: m.pellegrini@iit.cnr.it

E. Lastres
Sistemi Territoriali, via di Lupo Parra Sud 144, 56023 San Prospero, PI, Italy
e-mail: e.lastres@sister.it

F. Can et al. (eds.), *State of the Art Applications of Social Network Analysis*,
Lecture Notes in Social Networks, DOI: 10.1007/978-3-319-05912-9_3,

improves in speed and precision over current state of the art approximation methods. Tests conducted on a sample of publicly available large networks from the Stanford repository show that, for the exact algorithm, speed improvements of a factor ranging between 2 and 5 are possible on several such graphs, when the sparsity, measured by the ratio of tree-nodes to the total number of nodes, is in a medium range (30–50 %). For some large networks from the Stanford repository and for a sample of social networks provided by *Sistemi Territoriali* with high sparsity (80 % and above) tests show that our algorithm, named SPVB (for Shortest Path Vertex Betweenness), consistently runs between one and two orders of magnitude faster than the current state of the art exact algorithm.

1 Introduction

Social networks have demonstrated in the last few years to be a powerful and flexible concept useful to represent and analyze data emerging from social interactions and social activities. The study of these networks can thus provide a deeper understanding of many emergent social global phenomena. Moreover such analytic tools and concepts have been successfully adopted in a vast range of applications including communications, marketing and bioinformatics.

According to the standard paradigm of social networks, each agent/item is associated to a node of the network and the edges between pairs of nodes represent the relationship between them. Social networks are naturally represented as graphs, consequently graph theory and efficient graph algorithms play an important role in social network analysis. Among the analytic tools, *centrality indices* are often used to score (and rank) the nodes (or the edges) of the network to reflect their centrality position. The intuitive idea behind this class of indices is that a more central node is likely to be involved in many processes of the network, thus its importance increases.

Depending on what we mean with the word “important", different definitions of centrality are possible [1]. For example: degree centrality highlights nodes with a higher number of connections, closeness centrality highlights nodes easily reachable from other nodes, eigenvector centrality highlights nodes connected with authoritative nodes and betweenness centrality (BC) highlights nodes which are more likely to be information hubs. A complete compendium of many centrality definitions, problems and measures can be found in [2]. *Vertex betweenness* [3, 4] is one of the most broadly used centrality indices. The (vertex) betweenness of a vertex v is defined as the sum, for each pair of nodes (s, t) in the network, of the ratio between the number of shortest (aka geodesic) paths from s to t passing through v and the total number of shortest paths from s to t. The main assumption of this index is that the information flows in the network following shortest paths. Despite the fact that this assumption could be considered restrictive, betweenness finds a vast range of applications (e.g. in computing lethality for biological networks [5] and in bibliometry [6]).

A very similar concept, the *edge betweenness*, is defined in [3] where for an *edge* e, the sum is computed for each pair of nodes (s, t) of the ratio among the

number of shortest paths from s to t through the edge e over the number of all the shortest paths from s to t. Edge betweenness has a prominent application as a subroutine in the algorithm of Newman and Girvan [7] for community detection of complex networks. In this chapter, for sake of clarity, we discuss only the problem of computing efficiently vertex betweenness, however with minor modifications our approach applies to edge betweenness as well (see [8]). The computation of the betweenness centrality index is demanding because, for a given node v, all the shortest paths between each couple of nodes passing through v have to be counted (even if it is not necessary to explicitly enumerate them). This means that, in general, for fairly large networks the computation of this index based on a direct application of its definition becomes impractical, having complexity $O(n^3)$, for a graph with n nodes. Since the last decade the number and size of social networks have been consistently increasing over time, efficient algorithms have emerged to cope with this trend.

The fastest exact algorithm to date is due to Brandes [9]. It requires $O(n+m)$ space and $O(nm)$ time where n is the number of nodes and m the number of edges in the graph. For sparse graphs, where $m = O(n)$, Brandes' method is a huge improvement over the naive direct method, however it is still quadratic in n, regardless of any other special feature the input graph may have.

In this chapter we propose an evolution of the Brandes' algorithm, named SPVB (for Shortest Path Vertex Betweenness), which exploits some widespread topological characteristic of social networks to speed up the computation of the betweenness centrality index. We show that for nodes in the graph that belong to certain tree structures the beteenness value can be computed by a straightforward counting argument. The advantage of our approach is two-fold: on the one hand we do not need to count shortest paths for the subset of network nodes that have the required tree-structure, and, on the other hand, for the residual nodes we compute the shortest paths only between nodes belonging to the residual of the original graph, thus more efficiently. Our algorithm performance strictly depends on the number of nodes for which we can algebraically derive the betweenness. Therefore it works well in practice for social networks since we observed that such tree structures are quite frequent in the context of social networks where the number of edges of the graph is of the same order of magnitude of the number of nodes. Note, however, that SPVB still reduces to the Brandes' algorithm in a strict worst case scenario.

We have tested graphs with up to 500 K nodes, which is a fair size for many applications. However in some applications (e.g. web graphs, telephone calls graphs) we face much larger graphs in the regions of millions of nodes, and we might want to trade off speed and precision in computing the Betweenness Centrality (BC). In this case approximating betweenness may be the strategy of choice. Thus we combine our algebraic computation with the sampling approach in [10] so to gain the benefits of both (see Sect. 6), obtaining the algorithm ASPVB (for Approximate Shortest Path Vertex Betweenness).

We tested our algorithm on a set of 18 social graphs of *Sistemi Territoriali* which is an ICT company with headquarters in Italy, specializing in Business Intelligence applications. These graphs coming from real applications are very large and very sparse, a property SPVB exploits to gain in efficiency. Compared to Brandes' method

we can gain orders of magnitudes (between one and two) in terms of computation time. We also tested SPVB on a set of 16 social graphs from the Stanford Large Network Dataset Collection. We obtained marginal improvements on seven cases, speed ups by a factor from 2 to 6 in six cases, and speedups by orders of magnitude in two cases. At the best of our knowledge this approach is novel.

The chapter is organized as follows. Section 2 gives a brief survey of related work, while Sect. 3 gives key insights from Brandes' methods. In Sect. 4 we describe our method in detail for exact computations. In Sect. 5 we give the experimental results for exact computations. In Sect. 6 we give the approximation algorithm and the corresponding experimental results.

2 Related work

Let $G = (V, E)$ be the graph associated to a social network, we denote as: σ_{st} the number of shortest paths starting from the node s and ending in t, $\sigma_{st}(v)$ the cardinality of the subset of geodesic paths from s to t passing through v. Betweenness centrality [4] measures, for a given vertex v, the fraction of all the possible shortest paths between pairs of nodes which pass through v. Formally betweenness centrality $B(v)$ is defined as:

$$B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

The practical application of centrality indices depends also on the scalability of the algorithm designed to compute them. Early exact algorithms have a complexity in the order of $O(n^3)$ [11], where n is the number of nodes. Thus the computation of betweenness by this direct approach becomes impractical for networks with just a few thousands nodes.

In 2001 Brandes [9] developed the asymptotically fastest exact algorithm to date, that exploits a recursive formula for computing partial betweenness indices efficiently. It requires $O(n+m)$ space and $O(nm)$ time where n is the number of nodes and m the number of edges in the graph. For sparse graphs, where $m = O(n)$, Brandes' method is a huge improvement over the naive direct method, allowing to tackle graphs with tens of thousands of nodes.

Given the importance of the index, and the increasing size of networks to be analyzed, several strategies for scaling up the computation have been pursued. Algorithms for parallel models of computations have been developed (see e.g. [12–14]).

A second strategy is to resort to approximations of the betweenness [15]. In [10] the authors describe an approximation algorithm based on adaptive sampling which reduces the number of shortest paths computations for vertices with high centrality. In [16] the authors present a framework that generalizes the Brandes' approach to approximate betweenness. In [17] the authors propose a definition of betweenness which takes into account paths up to a fixed length k.

Another important complexity reduction strategy was presented in [18] where ego-networks are used to approximate betweenness. A ego-network is a graph composed by a node, called *ego*, and by all the nodes, *alters*, connected to the *ego*. Thus if two nodes are not directly connected, there is only a possible alternative path which passes through the ego node. The authors have empirically shown over random generated networks that the betweenness of a node v is strongly correlated to that of the ego network associated to v.

In order to extend the use of betweenness centrality to a wider range of applications, many variants of this index were proposed in the literature. For example in [19] the betweenness definition is applied to dynamic graphs, while in [20] geodesic paths are replaced with random walks. Modularity properties of social networks are used in [21] to define a notion of *Community Inbetweenness*. In experiments this measure is shown to weakly correlate with standard BC for networks of high modularity.

In graphs that change dynamically or are built incrementally (e.g. in a streaming model) algorithms have been proposed that dynamically update the betweenness by detecting efficiently those nodes whose BC is affected by the graph update (see [22, 23]).

In this chapter we propose to use specific local structures abundant in many types of social graphs in order to speed up the exact computation of the betweenness index of each node by an adaptation of the exact algorithm due to Brandes.

An approach that exploits special structures in social graphs is advocated also a chapter by Puzis et al. [24] that appeared just after the preliminary conference version of this work [25]. In Puzis et al. [24] develop two algorithms for exact BC computation. The first algorithm is advantageous when many nodes that are structurally equivalent, that is when they have the same set of neighbors. In this case equivalent nodes can be contracted into a single node and a quotient graph is generated. The original Brandes' procedure is adapted to work on the quotient graph, while computing the BC relative to the original graph. Experiments in [24] show a speed up from 2 to 3 in several Autonomous Systems (AS) graphs, and from 2 to 6 in DBLP co-authors graphs. The second algorithm generates the bi-connected components of the input graph, computes partial BC independently for each bi-connected, and then combines the results of the single components to produce the BC with respect to the original graph. Combining the two algorithms it is shown a speed from 2 to 7 in the set of AS-graphs. The edges of the tree-like structures we exploit are bi-connected components of the input graph thus, our trees are a special case of the components considered in [24], however the code we propose are much simpler than the algorithm in [24], while attaining comparable speed ups in the tested as-graphs.

3 Background

In this section we give some key features of Brandes' algorithm, since it gives a background to our approach. This method is based on an accumulation technique where the betweenness of a node can be computed as the sum of the contributions of

all the shortest paths starting from each node of the graph taken in turns. Given three nodes $s, t, v \in V$, Brandes introduces the notion of *pair-dependency* of s and t on v as the fraction of all the shortest paths from s to t through v over those from s to t:

$$\delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}}$$

The betweenness centrality of the node v is obtained as the sum of the pair-dependency of each pair of nodes on v. To eliminate the direct computation of all the sums, Brandes introduces the *dependency* of a vertex s on v as:

$$\delta_{s\bullet}(v) = \sum_{t \in V} \delta_{st}(v) \tag{1}$$

Thus the betweenness centrality B, of node v is given by summing the dependencies from all source nodes:

$$B(v) = \sum_{s \in V} \delta_{s\bullet}(v)$$

Observation 1 *If a node v is a predecessor of w in a shortest path starting in s, then v is a predecessor also in any other shortest path starting from s and passing through w [9].*

Arguing form the observation 1, Eq. 1 can be rewritten as a recursive formula:

$$\delta_{s\bullet}(v) = \sum_{w : v \in P_s(w)} \frac{\sigma_{sv}}{\sigma_{sw}}(1 + \delta_{s\bullet}(w)), \tag{2}$$

where $P_s(w)$ is the set of direct predecessors of a certain node w in the shortest paths from s to w, encoded in a BFS (Breadth First Search) rooted DAG (Directed Acyclic Graph) form node s.

4 Our Algorithm: SPVB

Our algorithm algebraically computes the betweenness of nodes belonging to trees in the network obtained by iteratively removing nodes of degree 1. Afterwards we apply a modification of Brandes' algorithm [9] to compute the betweenness of the nodes in the residual graph.

A first trivial observation is that nodes with a single neighbor can be only shortest paths endpoints, thus their betweenness is equal to zero. Thus we would like to remove these nodes from the graph. However, these nodes by their presence influence the betweenness of their (unique) neighbors. In fact, such neighbor v works as a bridge to connect the node to the rest of the graph and all the shortest paths to (from) this

node pass through that unique neighbor. Our procedure computes the betweenness of a node v as the sum of the contribution of all nodes for which v is their unique direct neighbor.

Following this strategy, once the contribution of the nodes with degree 1 has been considered in the computation of the betweenness of their neighbors, they provide no more information, and can be virtually removed from the graph. The removal of the nodes with degree 1 from the graph, can cause that the degree of some other node becomes 1. Thus the previous considerations can be repeated on a new set of degree one nodes. When we iterate, however, we need also to record the number of nodes connected to each of the degree one nodes that were removed from the graph. This recursive procedure allows us to algebraically compute the betweenness of trees in the graph.

4.1 Algorithm Formalization and Description

We will assume the input G to be connected, in order to simplify the argument. If G is not connected, the argument can be repeated for each connected component separately. Let F be the set of nodes in $G = (V, E)$ that can be removed by iteratively delete nodes of degree 1, and their adjacent edge. We call the nodes in F the *tree-nodes*. Let $G' = (V', E')$ be the residual graph for the residuals set of node, with $V' = V \backslash F$. The set F induces a forest in G, moreover the root of each tree T_i of the forest is adjacent to a unique vertex in V'. Each node in F is a root to a sub-tree. Let $R_G(w, F)$ be the set of nodes of trees in F having w as their root-neighbor in G'. The formula for the betweenness of node $v \in V$ involves a summation over pairs of nodes $s, t \in V$. Thus we can split this summation into sub-summations involving different types of nodes, and provide different algorithms and formulae for each case.

Tree-nodes. Let u be a node in F, and let $v_1, \ldots, v_k$ be the children of u in the tree T_u, and let T_{v_i}, for $i = 1, \ldots k$, be the subtrees rooted at v_i. When s and t are in the same subtree T_{v_i}, then there is only one shortest path connecting them completely in T_{v_i} and this path does not contain u, thus the contribution to $B(u)$ is null. When s is in some tree T_{v_i}, and t is in the complement $(V \backslash \{u\}) \backslash T_{v_i}$, then each shortest path connecting them will contain u. Thus the contribution to the betweenness of u is given by the number of such pairs. We will compute such number of pairs incrementally interleaved with the computation of the set F by peeling away nodes of degree 1 from the graph. When at iteration j, we peel away node v_i we have recursively computed the value of $|T_{v_i}|$, and also for the node u the value $|R_G(u, F_j)|$ which is the sum of the sizes of trees T_{v_h}, for $h \in [1, \ldots k], i \neq k$ already peeled away in previous iterations. The number of new pairs to be added to $B(u)$ is:

$$|T_{v_i}| \times (|(V \backslash \{u\}) \backslash T_{v_i}| - |R_G(u, F_j)|).$$

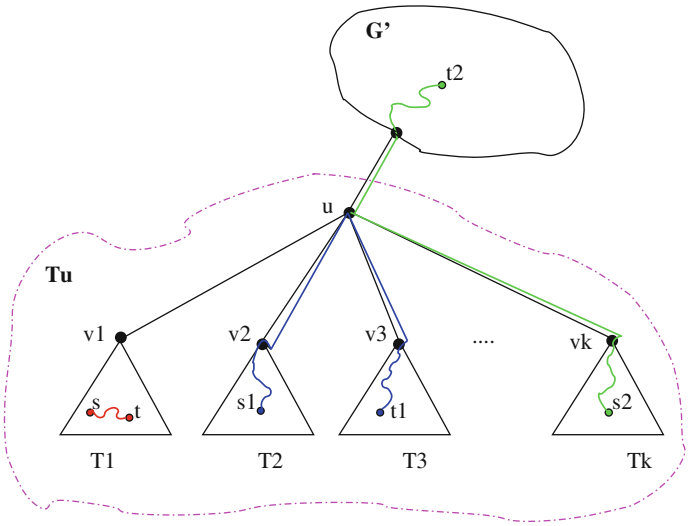


Fig. 1 Illustration of the tree-nodes structure and possible s–t paths involving tree nodes

This ensures that each pair (s,t) is counted only once. Finally observe that when both s and t are in V' no shortest path between them will contain u therefore their contribution to $B(u)$ is zero. Since the roles of s and t are symmetrical in the formula we need to multiply the final result by 2 in order to cont all pairs (s, t) correctly. The pseudocode for this procedure is shown in Sect. 4.2. See Fig. 1 for an illustration.

Residual graph nodes. Let u be a node in V', we will see how to modify Brandes' algorithm so that executing the modified version on the residual graph G' (thus at a reduced computational cost), but actually computing the betweenness of the nodes in $u \in V'$ relative to the initial graph G. Brandes algorithm's inner loop works by computing from a fixed node s a BFS search DAG in the input graph, which is a rooted DAG (rooted at s), and by applying a structural induction from the sinks of the DAG towards the root as in formula (2).

Subcase 1.If a node $x \in V'$ has $R(x, F) \neq \emptyset$ the tree formed by $R(x, F)$ and x would be part of the BFS DAG in G having its source in V', however, since we run the algorithm on the reduced graph G', we need to account for the contribution of the trimmed trees to the structural recursive formula (2). The correction term for $\delta_{s\bullet}(x)$ is equal to $|R_G(x, F)|$ since each shortest path from s to $y \in R_G(x, F)$ must contain x. Thus we obtain the new formula:

$$\delta_{s\bullet}(u) = \sum_{w:u \in P_s(w)} \frac{\sigma_{su}}{\sigma_{sw}} (1 + \delta_{s\bullet}(w) + |R_G(w, F)|))$$

Note that in the development of the above formula $R(s, F)$ does not appear. Since no shortest path from $s \in V'$ to any $t \in R(s, F)$ may include a node $u \in V'$, this subtree has zero contribution to $\delta_{s\bullet}(u)$.

Subcase 2. Consider now a node $x \in R(s, F)$ as source for the BFS. In the computation of $\delta_{x\bullet}(u)$, for $u \in V'$ each shortest path from x to $t \in R(s, F)$ cannot contain u thus gives zero contribution. For $t \in V \backslash R(s, F)$, such shortest path would contain a shortest path from s, thus we have $\delta_{x\bullet}(u) = \delta_{s\bullet}(u)$ for all $x \in R(s, F)$. In order to account for these contributions to $B(u)$ it suffices to multiply the contribution $\delta_{s\bullet}$ by $(1 + |R(s, F)|)$, obtaining:

$$B(u) = B(u) + \delta_{s\bullet}(u) * (1 + R_G(s, F)).$$

4.2 Algorithm Pseudo-Code

In the following Algorithm 1 we show the pseudo-code for SPVB (Shortest-paths vertex betweenness) preprocessing, handling degree 1 nodes. For simplicity we assume G to be connected. For a disconnected graph G, the algorithm should be applied to each connected component separately. For a node v of degree 1 at a certain stage of the iteration, the vector p records the number of nodes in a subtree rooted at v (excluding the root). For any other node u, vector p records the sum of the sizes of subtrees rooted at children of that node that have been deleted in previous iterations.

SPVB:
Data: undirected unweighted graph G=(V,E)
Result: the graph's node betweenness $B[v]$ for all $v \in V$
$B[v] = 0, v \in V$; $p[v] = 0, v \in V$; $i = 0$;
$G^i = G$; $deg_1 = \{v \in V^i | deg(v) = 1\}$;
repeat
  $v \leftarrow deg_1$;
  $u \in V^i.(v, u) \in E^i$;
  $B[u] = B[u] + 2(n - p[v] - p[u] - 2)(p[v] + 1)$;
  remove v from deg_1;
  $p[u] = p[u] + p[v] + 1$;
  $i++$;
  $V^i = V^{i-1} \backslash \{v\}$
  $E^i = E^{i-1} \backslash \{(v, u)\}$
  if $deg(u) = 1$ **then** $u \rightarrow deg_1$; /*$deg(u)$ `is computed on the new graph` G^i */
until $deg_1 = \emptyset$;
if $|V^i| > 1$ **then**
  Brandes_modified(G^i, p, B)
end

Algorithm 1: Shortest-paths vertex betweenness

The modification of Brandes' algorithm does not change its asymptotic complexity, which however must be evaluated on the residual graph with $n' = |V| - |F|$ nodes and $m' = |E| - |F|$ edges, thus with a time complexity $O(n'm')$. The complexity of the first part of SPVB is constant for each node in F, except for the operations needed to dynamically modify the graph G^i in Algorithm 1 and maintain the set of degree-1 nodes. With standard dynamic dictionary data structure we have an overhead of $O(\log n)$ for each update operation.

5 Experiments

In order to evaluate the performance of our algorithm we run a set of experiments using both a collection of 18 graphs provided by *Sistemi Territoriali (SisTer)*, which is an Italian ICT company involved in the field of data analysis for Business Intelligence and a collection of graphs downloaded from the Stanford Large Network Dataset Collection.[1] Since both SPVB and Brandes' compute the exact value of betweenness, we tested the correctness of the implementation by comparing the two output vectors. Here we report only on the running time of the two algorithms. For our experiments we used a standard PC endowed with a 2.5 GHz Intel Core 2, 8 Gb of RAM and Linux 2.6.37 operating system. The two algorithms were implemented in Java. In order to avoid possible biases in the running time evaluation due to the particular CPU architecture, we decided to implement the algorithm as a mono-processor sequential program.

SisTer Collection. In Table 1 we report the graph id, the number of nodes and edges in the SisTer collection and the percentage of tree-nodes in each graph. Note that a very large percentage of the nodes can be dealt with algebraically by SPVB and the residual graph, on which we ran a modified Brandes', is quite small relative to the original size.

Figure 2 compares the running time of our and Brandes' algorithms. On the x-axis we report the graph id, while on the y-axis we report in logarithmic scale the running time expressed in seconds. From Fig. 2 it is possible to observe that SPVB is always more than one order of magnitude faster than the procedure of Brandes, sometimes even two orders of magnitude faster. For graph G1, with 233,377 nodes, for example, we were able to finish the computation within 1 h while Brandes' needs approximately two days. For graph G6, with 169,059 nodes, we could complete in about 1 min, compared to two days for Brandes. A notable result is shown also for graph G18 which is our biggest in this collection. In this case SPVB required approximately 2, 4 days to finish while Brandes' could not terminate in one month (data not shown).

Stanford Collection. We have selected a subset of graphs from the Stanford collection, using the following criteria. First the graphs have been ranked by number of

[1] http://snap.stanford.edu/data/

Brandes_modified:
Data: directed graph $G = (V, E)$,
for each v:
the number of tree-nodes connected to v: $p[v]$,
the partial betweenness computed for v: $B[v]$
Result: the graph's node betweenness $B[v]$
for $s \in V$ **do**
  S = empty stack;
  P[w]= empty list,$w \in V$;
  $\sigma[t] = 0, t \in V$;$\sigma[s] = 1$;
  $d[t] = -1, t \in V^i$; $d[s]$ =0;
  Q= empty queue;
  enqueue $s \rightarrow Q$;
  while *Q not empty* **do**
    dequeue $v \leftarrow Q$;
    push $v \rightarrow S$;
    forall *neighbor w of v* **do**
      // w found for the first time?
      if $d[w] < 0$ **then**
        enqueue $w \rightarrow$ Q;
        d[w]=d[v] + 1;
      end
      // shortest path to w via v?
      if $d[w] = d[v] + 1$ **then**
        $\sigma[w] = \sigma[w] + \sigma[v]$;
        append $v \rightarrow P[w]$;
      end
    end
  end
  $\delta[v] = 0, v \in V$;
  // S returns vertices in order of non-increasing distance from s
  while *S not empty* **do**
    pop w$\leftarrow$ S;
    for $v \in P[w]$ **do**
      $\delta[v] = \delta[v] + \frac{\sigma[v]}{\sigma[w]}(\delta[w] + p[w] + 1)$;
    end
    if $w \neq s$ **then**
      $B[w] = B[w] + \delta[w] \times (p[s] + 1)$
    end
  end
end

Algorithm 2: Modified Brandes' algorithm

nodes and we have selected representative graphs from as many categories as possible (Social networks, Communication Networks, Citation networks, Collaboration networks, Web graphs, Internet peer-to-peer networks, and Autonomous systems graphs). We have excluded graphs that because of their size would take more than one week of computing time. In Table 2 we have listed these graphs, their size in number of nodes and edges, and the percentage of tree-nodes, which is the most important parameter influencing the performance of our method. Each input graph was considered undirected. We decided a cut-off time of seven days. In order to

Table 1 SisTer Collection. For each graph it is listed the number of nodes, the number of edges, and the percentage of tree-nodes. The graphs need not be connected

Graph ID	Node #	Edge #	Tree nodes (%)
G1	233,377	238,741	86
G2	14,991	14,990	99
G3	15,044	15,101	85
G4	16,723	16,760	84
G5	16,732	16,769	84
G6	169,059	169,080	99
G7	16,968	17,026	84
G8	3,214	3,423	95
G9	3,507	3,620	96
G10	3,507	3,620	96
G11	3,519	3,632	96
G12	44,550	46,519	77
G13	46,331	46,331	99
G14	47,784	48,461	84
G15	5,023	5,049	93
G16	52,143	53,603	85
G17	8,856	10,087	89
G18	506,900	587,529	80

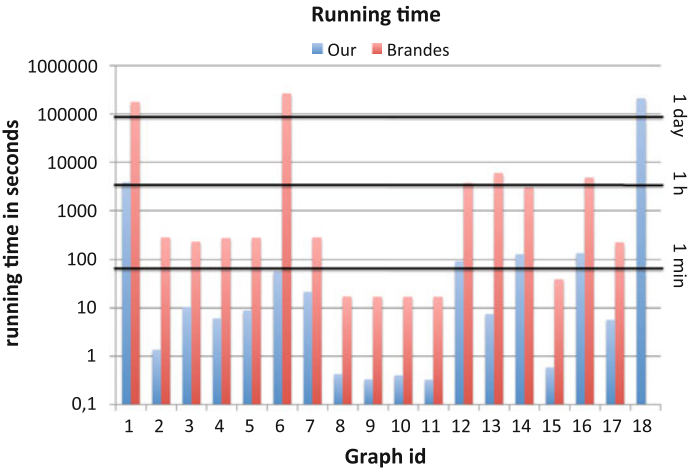


Fig. 2 A comparison of the running time of our algorithm SPVB (*left*) and Brandes' (*right*) on 18 sparse large graphs. The ordinate axis report running time in seconds and is in logarithmic scale. Data for Brandes on graph 18 is missing due to time-out

measure the convergence of the two methods we collected also the partial output of the two algorithms every 24 h of execution. In Table 3 the running time, expressed in seconds, of the two methods is shown, and the speed up factor. As it is expected the speed up factor is strongly correlated to the fraction of the tree-nodes in the graph.

Table 2 Selected graphs from the Stanford Collection

Graph name	Node #	Edge #	Tree nodes (%)
ca-GrQc	5,242	28,980	21
as20000102	6,474	13,233	**36**
ca-HepTh	9,877	51,971	20
ca-HepPh	12,008	237,010	11
ca-AstroPh	18,772	396,160	6
ca-CondMat	23,133	186,936	9
as-caida20071112	26,389	106,762	**38**
cit-HepTh	27,770	352,807	5
cit-HepPh	34,546	421,578	4
p2p-Gnutella31	62,586	147,892	**46**
soc-epinion1	75,879	508,837	**51**
soc-sign-Slashdot090221	82,144	549,202	**36**
soc-Slashdot0922	82,168	948,464	2
soc-sign-epinions	131,828	841,372	**51**
Email-EuAll	265,214	420,045	**80**
web-NotreDame	325,729	1,497,134	**51**

For each graph it is listed the number of nodes, the number of edges, and the percentage of tree-nodes, which is the most important parameter affecting the time performance
In bold are marked data sets with percentage of tree-nodes above 30 %

We notice a speed-up factor ranging from 2 to almost 6 when the ratio of tree-nodes to the total number of nodes is in the range 30–50 %.

Two large test graphs are quite noticeable. Graph *Email-EuAll* has a percentage of 80 % of tree-nodes which is a value closer to those found in the SisTer collection, thus the speed up measured is at least 27 (since we stopped Brandes' after one week). That value is between one and two orders of magnitude, consistently with those measured in the SisTer collection.

For the *web-NotreDame* graph, which is the largest graph in our sample of the Stanford collection, we estimate the convergence properties of the two algorithms as follows. SPVB has been run to completion (in about 9 days) in order to have the exact target solution vector. Also at fixed intervals each day we recorded the intermediate values of the betweenness vectors for both algorithms. For each vertex we compute the ratio of the intermediate value over the target value (setting $0/0$ to value 1), and then we average over all the vertices. This measure is strongly biased by the fact that for leaves (nodes with degree 1) both Brandes and SPVB assign at initialization the correct value 0, thus in this case precision is attained by default. To avoid this bias we repeat the measurement by averaging only over those nodes with final value of betweenness greater than zero (see Fig. 3). From Fig. 3 we can appreciate that the average convergence rate is almost linear in both case, but the curve for SPVB has a much higher slope. After 7 days our algorithm reached about 75 % of the target, against 10 % of Brandes', by a linear extrapolation we can thus predict a speed up factor of about 8.

Table 3 Running time (in seconds) of the two methods over selected Stanford collection graphs, and their ratio (speed up factor)

Graph name	Node #	Brandes (s)	SPVB (s)	Ratio
ca-GrQc	5,242	35 s	24 s	1.45
as20000102	6,474	141 s	54 s	**2.65**
ca-HepTh	9,877	230 s	148 s	1.55
ca-HepPh	12,008	703 s	563 s	1.24
ca-AstroPh	18,772	2,703 s	2,447 s	1.10
ca-CondMat	23,133	3,288 s	2,718 s	1.21
as-caida20071112	26,389	6,740 s	2,014 s	**3.34**
cit-HepTh	27,770	8,875 s	8,227 s	1.07
cit-HepPh	34,546	16,765 s	15,636 s	1.07
p2p-Gnutella31	62,586	74,096 s	15,573 s	**4.76**
soc-Epinion1	75,879	145,350 s	25,771 s	**5.64**
soc-sign-Slashdot090221	82,140	199,773 s	64,905 s	**3.07**
soc-Slashdot0902	82,168	199,544 s	190,536 s	1.04
soc-sign-epinions	131,828	564,343 s	96,738 s	**5.83**
Email-EuAll	265,214	>7 days	22,057 s	>**27**
web-NotreDame	325,729	–	≈9 days	≈**8**

In bold are marked data sets with a performance ratio above 2

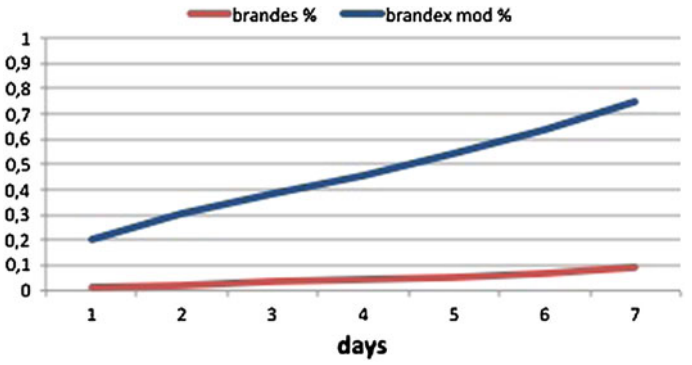


Fig. 3 Evolution in time of the average (over the vertices) ratio of the partial betweenness values over the final betweenness value. In the averaging leaves are excluded

6 Approximating Betweenness Centrality

In this section we show how we can combine our algebraic approach to computing BC with the approximation scheme in [10], which is based on adaptive sampling. First of all we notice that it is not possible in general to choose a random sample size for each data set that ensures a uniform relative error ϵ at each node. In [10] it is shown that with high probability, we can approximate the betweeenness $B[v]$ of a node v in a graph of n nodes, up to a factor $1/\epsilon$, with a number s of randomly chosen source nodes (from here referred as *pivots*), where $s = s(B[v], n, \epsilon)$. Since s depends

Table 4 Running time (in seconds) of the two approximate methods over selected Stanford collection graphs, their mean relative error and their ratio (speed up factor)

Graph name	Node #	MRE Approx			Time Approx		
		ASPVB	Brandes	Ratio	ASPVB (s)	Brandes (s)	Ratio
ca-GrQc	5,242	**0.260**	0.374	**1.43**	5.359	11.475	**2.12**
as20000102	6,474	**0.394**	0.427	1.08	5.709	8.058	1.41
ca-HepTh	9,877	**0.329**	0.457	**1.38**	13.479	23.322	**1.73**
ca-HepPh	12,008	**0.353**	0.472	**1.34**	29.881	48.448	1.62
ca-AstroPh	18,772	**0.413**	0.548	**1.32**	83.516	100.566	1.20
ca-CondMat	23,133	**0.341**	0.458	**1.34**	89.589	90.286	1.01
as-caida20071112	26,389	**0.435**	0.454	1.04	74.025	126.258	**1.70**
cit-HepTh	27,770	**0.729**	0.742	1.01	209.085	211.766	1.01
cit-HepPh	34,546	0.724	**0.246**	0.34	330.874	347.646	1.05
p2p-Gnutella31	62,586	**0.362**	0.537	**1.48**	392.815	892.982	**2.27**
soc-Epinion1	75,879	**0.398**	0.466	1.17	650.055	1,586.527	**2.44**
soc-sign-Slashdot090221	82,140	**0.566**	0.595	1.05	1,154.123	2,111.585	**1.82**
soc-Slashdot0902	82,168	0.616	**0.604**	0.98	2,003.166	2,081.609	1.03
soc-sign-epinions	131,828	**0.566**	0.595	1.05	1,154.123	2,111.585	**1.83**
Email-EuAll	265,214	0.072	**0.067**	0.93	868.456	25,704.993	**29.59**
web-NotreDame	325,729	0.671	**0.539**	0.80	14,364.103	51,372.872	**3.57**

Each value is the mean of 10 runs with different random samples
For Columns ASPVB and Brandes the bold values indicates the smallest MRE value among the two approximate methods
For the column "Ratio" bold values indicate values above 1.30 in improved error performance
For the Time approximation ratio bold values indicate a performance ratio above 1.70

also on the value $B[v]$ we cannot hope to have a uniform approximation factor bound over all the nodes in the graph. For this reason, we select an uniform sample size function having as input only the number of nodes and we *measure* the resulting mean relative error in each experiment. Thus we select a fixed value $s = \sqrt{n}$, and we measure empirically the mean relative error against the exact value.[2] The BC value of tree-nodes is known exactly and their contribution to the BC value of other nodes can be attributed to the root of the tree, therefore we restrict the sampling on the nodes in the residual graph. Also the shortest path computations are done in the residual graph. Note however that the expansion factor used to estimate the BC is referred to the size of the original graph. The pseudocode of the combined approach is shown in Algorithms 3, 4, and 5.

6.1 Approximate Algorithm Pseudo Code

In the following Algorithm 3 we show the pseudo-code for ASPVB (Approximate Shortest-paths vertex betweenness) preprocessing. For the sake of clarity we consider

[2] For nodes whose BC exact value is zero, the partial BC contribution for any source is also zero, thus the sampling procedure will estimate the correct value, zero.

G to be connected. For disconnected graphs the same procedure should be applied to each component.

ASPVB:
Data: unweighted graph G=(V,E)
Result: the graph's node approximate betweenness $c_B[v]$ for all $v \in V$
$c_B[v] = 0, v \in V$;
$c_0[v] = 0, v \in V$; /*$c_0[v]$ stores the algebraic computation of the degree one nodes */
$p[v] = 0, v \in V; i = 0$;
$G^i = G$;
$deg_1 = \{v \in V^i | deg(v) = 1\}$;
repeat
  $v \leftarrow deg_1$;
  $u \in V^i.(v, u) \in E^i$;
  $c_0[u] = c_0[u] + 2(n - p[v] - p[u] - 2)(p[v] + 1)$;
  remove v from deg_1;
  $p[u] = p[u] + p[v] + 1$;
  $i + +$;
  $V^i = V^{i-1} \backslash \{v\}$
  $E^i = E^{i-1} \backslash \{(v, u)\}$
  if $deg(u) = 1$ **then** $u \rightarrow deg_1$; /*$deg(u)$ is computed on the new graph G^i) */
until $deg_1 = \emptyset$;
if $|V^i| > 1$ **then**
  ComputeApproximateBetweenness(G^i, p, $c_B[v]$, $c_0[v]$, $|V|$)
end
else
  $c_B[v] = c_0[v]$
end

Algorithm 3: Approximate shortest-paths vertex betweenness

The algorithm takes as input an undirected graph $G = (V, E)$ and returns the approximate betweenness value for each node of the graph. Since the algebraic computation is the same of the exact algorithm, for nodes whose betweenness is algebraically computed the returned value is exact.

In Algorithm 4 we show our approximate algorithm for the residual graph. We compute the betweenness of the nodes in each path starting from a generic node s as if we were considering the path in the whole graph (see lines 1 and 2 in Algorithm 4). This is because we need to consider the contribution of the node within the whole graph when computing its approximate value. We maintain update an auxiliary structure (see line 3 in Algorithm 4) with the weight of each node in the shortest path from s for all the nodes connected to the residual graph through s. This value will be used in case of exact computation (see line 1 in Algorithm 5) to return the exact value of each node. As in [10], the computation of the approximate betweenness is the sum of the contributions due to the pivots times $\sqrt{n}$.

Table 5 Running time (in seconds) of the two approximate methods over selected Sister Collection graphs, their mean relative error and their ratio (speed up factor)

Graph name	Node #	MRE Approx			Time Approx		
		ASPVB	Brandes	Ratio	ASPVB (s)	Brandes (s)	Ratio
G8	3,214	**0.166**	0.272	**1.63**	0.669	2.073	**3.09**
G9	3,507	**0.222**	0.251	1.13	0.715	2.260	**3.16**
G10	3,507	0.250	**0.236**	0.94	0.687	2.161	**3.14**
G11	3,519	0.271	**0.236**	0.87	0.690	2.033	**2.94**
G15	5,023	**0.075**	0.347	**4.63**	0.912	3.750	**4.11**
G17	8,856	**0.168**	0.402	**2.39**	2.802	9.517	**3.39**
G2	14,991	**0.000**	0.023	–	–	13.988	**–**
G3	15,044	**0.022**	0.229	**10.4**	4.151	12.863	**3.09**
G4	16,723	**0.017**	0.159	**9.30**	3.607	14.440	**4.00**
G5	16,732	**0.019**	0.159	**8.36**	3.704	14.554	**3.92**
G7	16,968	**0.028**	0.158	**5.64**	5.104	14.736	**2.88**
G12	44,550	**0.050**	0.323	**6.46**	17.007	99.715	**5.86**
G13	46,331	0.070	**0.016**	0.22	5.377	130.774	**24.32**
G14	47,784	**0.028**	0.231	**8.25**	20.658	108.105	**5.23**
G16	52,143	**0.035**	0.235	**6.71**	22.431	131.889	**5.87**
G6	169,059	0.120	**0.001**	**120.00**	57.238	2,156.538	**37.67**
G1	233,377	**0.049**	0.264	**5.38**	338.383	2,461.949	**7.27**
G18	506,900	**0.166**	0.366	**2.20**	4,849.750	160,623.840	**33.12**

Each value is the mean of 10 runs with different random samples. For G2 the sample size is the residual graph size, thus the computation is exact
For Columns ASPVB and Brandes the bold values indicates the smallest MRE value among the two approximate methods
For the column "Ratio" bold values indicate values above 1.30 in improved error performance
For the Time approximation ratio bold values indicate a performance ratio above 1.70

6.2 Experimental Results on Approximating Betweenness

In Tables 4 and 5 we report quality (measured by the mean relative error) versus time measurements over ten runs of our approximation algorithm and the original scheme in [10], where both algorithms are executed with the same number of samples.

We notice that almost always on the graphs from the Stanford repository our combined approximations scheme gains against [10] in quality (reducing the mean relative error), even with a low percentage of tree-nodes. We also gain in speed by a factor between 3.5 and 1.7 for graphs with a large percentage of tree-nodes. The speedup factor is not as high as in the exact case since the uniform sampling size (same number of sources) eliminates one of the gain factors we have in the exact case. For the Sister Collection, due to the very high sparsity we gain substantially in speed (by a factor 3 or larger), and the error is reduced (often by an order of magnitude) in 14 tests over 18. In two cases, $G6$ and $G13$, the speed up effect is large, but the quality measure is worse. This is due to the fact that the sample size is smaller but close to the residual graph size, thus the final scaling factor introduces a small bias. However in such cases the exact algorithm of Sect. 4, should be run, as there is no time gain in resorting to the approximated version.

ComputeApproximateBetweenness:
Data: directed graph $G = (V, E)$,
for each v:
the number of tree-nodes connected to v: $p[v]$,
the accumulator for the approximate betweenness$v$: $AB[v]$,
the betweenness algebraically computed so far v: $c_0[v]$,
the number of nodes in the original graph n
Result: the graph's node approximate betweenness $AB[v]$
pivot_number = 0;
$AB_s[v] = 0, v \in V$
max = sqrt(n)
if $max > |V|$ **then**
  $max = |V|$
end
while $pivot\_number < max$ **do**
  $pivot\_number++$
  pivot = choose($n \in V$)
  s = pivot
  S = empty stack;
  P[w]= empty list, $w \in V$;
  $\sigma[t] = 0, t \in V$;$\sigma[s] = 1$;
  $d[t] = -1, t \in V^i$; $d[s]$ =0;
  Q= empty queue;
  enqueue $s \rightarrow Q$;
  while *Q not empty* **do**
    dequeue $v \leftarrow Q$;
    push $v \rightarrow S$;
    forall *neighbor w of v* **do**
      // w found for the first time?
      if $d[w] < 0$ **then**
        enqueue $w \rightarrow$ Q;
        d[w]=d[v] + 1;
      end
      // shortest path to w via v?
      if $d[w] = d[v] + 1$ **then**
        $\sigma[w] = \sigma[w] + \sigma[v]$;
        append $v \rightarrow P[w]$;
      end
    end
  end
  $\delta[v] = 0, v \in V$;
  // S returns vertices in order of non-increasing distance from s
  while *S not empty* **do**
    pop w$\leftarrow$ S;
1   $\delta[w] = \delta[w] + p[w]$
    for $v \in P[w]$ **do**
2     $\delta[v] = \delta[v] + \frac{\sigma[v]}{\sigma[w]}(\delta[w] + 1)$
      if $w \neq s$ **then**
        $AB[w] = AB[w] + \delta[w]$
3       $AB_s[w] = AB[w] + (\delta[w] * p[w])$
      end
    end
  end
end
ApproximateValue($AB, AB_s, c_0, n, max, |V|$)

Algorithm 4: Modified Brandes' algorithm

ApproximateValue:
Data: for each v:
Approximate betweenness AB,
the betweenness value depending on nodes not in the residual graph AB_s,
the algebraic betweenness computation c_0,
the number of nodes in the original graph, n
the number of pivot, max
the number of nodes in the residual graph, nr
Result: the graph's node approximate betweenness $AB[v]$
i=0;
1 **if** $max = nr$ **then**
  for $i < n$ **do**
    $AB[i] = AB[i] + AB_s[i] + C_0[i]$
  end
else
  for $i < n$ **do**
    if $AB[i] \neq 0$ **then**
      $AB[i] = AB[i] * \frac{n}{max}$
    else
      $AB[i] = c_0[i]$
    end
  end
end

Algorithm 5: Rescaling of the results.

7 Conclusions

Brandes' algorithm for computing betweenness centrality in a graph is a key breakthrough beyond the naive cubic method that computes explicitly the shortest paths in a graph. However, it is not able to exploit possible additional locally sparse features of the input graph to speed up further the computation on large graphs. In this work we show that combining exact algebraic determination of betweenness centrality for some tree-like sub-graphs of the input graph, with a modified Brands' procedure on the residual graph we can gain orders of magnitudes (between one and two) in terms of computation time for very sparse graphs, and a good factor from 2 to 5, in moderately sparse graphs. Also in the approximate setting combining the algebraic technique with an adaptive sampling our experiments show gains in speed and/or precision over state of the art approximate algorithms. At the best of our knowledge this approach is novel. Among the graphs tested in this chapter, we did not find a significant number of tree-nodes only in author collaboration graphs and citation graphs, while for the other categories we found a significant number of tree-nodes. We thus conjecture that this feature is common enough in a range of social networks so to make the application of our method an interesting option when exact or approximate betweenness is to be computed.

As future work we plan to explore further this approach by determining other classes of subgraphs (besides trees) in which we can gain by the direct algebraic

determination of the betweenness. Moreover the impact of our approach combined with other approximation schemes will be investigated.

Acknowledgments This research is partially supported by the project *BINet* "Nuova Piattaforma di Business Intelligence Basata sulle Reti Sociali" funded by Regione Toscana POR CReO 2007–2013 Programme.

References

1. Koschatzki D, Lehmann K, Peeters L, Richter S, Tenfelde-Podehl D, Zlotowski O (2005) Centrality indices. In: Brandes U, Erlebach T (eds) Network analysis. Lecture notes in computer science, vol 3418. Springer, Berlin, pp 16–61
2. Borgatti SP (2005) Centrality and network flow. Social Netw 27(1):55–71
3. Anthonisse JM (1971) The rush in a directed graph. Technical Report BN 9/71, Stichting Mathematisch Centrum, 2e Boerhaavestraat 49 Amsterdam
4. Freeman LC (1977) A set of measures of centrality based on betweenness. Sociometry 40(1):35–41
5. Del Sol A, Fujihashi H, O'Meara P (2005) Topology of small-world networks of protein–protein complex structures. Bioinformatics 21:1311–1315
6. Leydesdorff L (2007) Betweenness centrality as an indicator of the interdisciplinarity of scientific journals. J Am Soc Inf Sci Technol 58:1303–1309
7. Girvan M, Newman MEJ (2002) Community structure in social and biological networks. Proc Natl Acad Sci USA 99:7821–7826
8. Brandes U (2008) On variants of shortest-path betweenness centrality and their generic computation. Social Netw 30(2):136–145
9. Brandes Ulrik (2001) A faster algorithm for betweenness centrality. J Math Sociol 25(2): 163–177
10. Bader D, Kintali S, Madduri K, Mihail M (2007) Approximating betweenness centrality. In: Bonato A, Chung F (eds) Algorithms and models for the Web-Graph, vol 4863. Lecture Notes in Computer Science. Springer, Berlin, pp 124–137
11. Jacob R, Dirk K, Lehmann K, Peeters L, Tenfelde-Podehl D (2005) Algorithms for centrality indices. In: Brandes U, Erlebach T (eds) Network analysis. Lecture notes in computer science, vol 3418. Springer, Berlin/Heidelberg, pp 62–82
12. Bader DA, Madduri K (2006) Parallel algorithms for evaluating centrality indices in real-world networks. In: International conference on parallel processing, 2006, ICPP 2006, pp 539–550
13. Kintali S (2008) Betweenness centrality: algorithms and lower bounds. CoRR, abs/0809.1906
14. Madduri K, Ediger D, Jiang K, Bader DA, Chavarria-Miranda D (2009) A faster parallel algorithm and efficient multithreaded implementations for evaluating betweenness centrality on massive datasets. Parallel and distributed processing symposium, international, pp 1–8
15. Brandes U, Pich C (2007) Centrality estimation in large networks. I J Bifurcat Chaos 17(7):2303–2318
16. Geisberger R, Sanders P, Schultes D (2008) Better approximation of betweenness centrality. In: ALENEX, pp 90–100
17. White S, Smyth P (2003) Algorithms for estimating relative importance in networks. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '03, ACM, New York, pp 266–275
18. Everett M, Borgatti SP (2005) Ego network betweenness. Social Netw 27(1):31–38
19. Carpenter T, Karakosta G, Shallcross D (2002) Practical issues and algorithms for analyzing terrorist networks, 2002. Invited paper at WMC 2002
20. Newman MEJ (2005) A measure of betweenness centrality based on random walks. Social Netw 27(1):39–54

21. Chan SY, Leung IXY, Liò P (2009) Fast centrality approximation in modular networks. In: CIKM-CNIKM, pp 31–38
22. Green O, McColl R, Bader DA (2012) Fast algorithm for incremental betweenness centrality. In: Proceeding of SE/IEEE international conference on social computing (SocialCom), 3–5 Sept 2012
23. Lee M-J, Lee J, Park JY, Choi RH, Chung C-W (2012) QUBE: a quick algorithm for updating betweenness centrality. In: Proceedings of the 21st international conference on World Wide Web, WWW '12, ACM, New York, pp 351–360
24. Puzis R, Zilberman P, Elovici Y, Dolev S, Brandes U (2012) Heuristics for speeding up betweenness centrality computation. In: Proceeding of SE/IEEE international conference on social computing (SocialCom), 3–5 Sept 2012
25. Baglioni M, Geraci F, Pellegrini M, Lastres E (2012) Fast exact computation of betweenness centrality in social networks. In: Proceedings of the 2012 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM 2012), Istambul, Turkey, 26–29 Aug 2012

An Agent-Based Modeling Framework for Social Network Simulation

Enrico Franchi

Abstract Agent-based modeling has been frequently adopted as a research tool in the fields of social and political sciences. Although recently social network analysis has generated a new wave of interest in many different research fields, nonetheless software instruments specifically created for agent-based social network simulation are still missing. However, restricting the field of interest specifically to social network models and simulations instead of supporting general agent-based ones, allows for the creation of easier to use, more focused tools. In this work, we propose an agent-based modeling framework for simulations over social networks. The models are written in a purposely developed, domain-specific language that helps in mapping social-network concepts to agent-based ones. Our framework is created to deal with large simulations and to work effortlessly with other social network analysis toolkits.

1 Introduction

In the last 10 years, the pervasive adoption of social networking sites has deeply changed the web and such sites became an unprecedented social phenomenon [10]. According to some recent studies, web sites have attracted users with very weak interest in technology, including people who, before the social networking revolution, were not even regular users of either popular Internet services or computers in general [44].

In the preceding decade, Agent-Based Modeling (ABM) was widely adopted as a research tool in the fields of social and political sciences, because a multi-agent system is a suitable means for modeling and performing simulations on complex

E. Franchi (✉)
Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Parma, Parma, Italy
e-mail: efranchi@ce.unipr.it

F. Can et al. (eds.), *State of the Art Applications of Social Network Analysis*, Lecture Notes in Social Networks, DOI: 10.1007/978-3-319-05912-9_4,

systems: a model consists of a set of agents whose execution emulates the behavior of the various individuals that constitute the system [46].

Seminal research in this direction was due to Epstein and Axtel and led to the development of Sugarscape [14], which builds upon the earlier ideas of cellular automata [47], and more specifically of Conway's Game of Life [21], by adding social elements, agent's individual features, genetics and different types of environment in which the agents are situated.

Multi-agent systems are especially appropriate to model systems (1) where complex features arise from repeated relatively simple interactions, and (2) which are dominated by discrete decisions. In particular, starting from Sugarscape, ABM gave important results in social science because it represents and simulates not only the behavior of individuals or groups of individuals but also their interactions that combine to create to the emergent behavior [2, 13].

In parallel with the theoretical development of ABM a number of software platforms were developed to ease the task of running the simulations; among these the most popular are Swarm [34], Mason [29], RePast [35] and NetLogo [45], which, however are not specifically tailored for social network simulations.

In this chapter, instead, we introduce a different kind of ABM tool specifically created for network generation and general processes over networks. The tool we propose does not aim at being a general purpose agent-based modeling tool, thus remaining a relatively simple software system, whereas it is extensible where it really matters (e.g., supporting different representations for networks, from simple memory-based ones to pure disk-based storage for huge simulations).

Simulation parameters can be easily set both programmatically or as command line arguments, so that simulations can be easily run with different starting parameters, for example to perform sensitivity analysis or any kind of batch execution.

People with little programming skills shall be able to specify and run their simulations, but skilled programmers must not feel limited. Consequently, we designed an agent-based Domain-Specific Language (DSL) created to express and execute (1) social network generation models and (2) general processes over social networks. The system focuses on expressing communicative acts among the nodes in the network, as they can be used to express any process conveniently (e.g., link creation/removal or information/disease diffusion).

In Sect. 2 we more thoroughly expose the motivations for this work and we compare it with the ABM tools mentioned before. In Sect. 3 we describe the logical metamodel of the simulation toolkit, and in Sects. 4 and 5 we present the DSL design and the concurrency model, respectively. We also show how some well known models are implemented in our toolkit. Finally, in Sect. 6 we draw some conclusions.

2 Motivations of the Work

At the beginning of this section we briefly introduce ABM, and, specifically, the general structure of an agent-based simulation; then, in Sect. 2.2, we put ABM in the perspective of social network modeling, i.e., we present the unique traits of social

network simulations and how they impact on designing an ABM of social network processes. Finally, we discuss the state of the art of ABM and how it is related to our project.

2.1 Agent-Based Simulation Fundamentals

An agent-based simulation has three main components: (a) a set of *agents*, (b) a set of *agent relationships* that determines a *topology* and (c) the *agents environment* [31].

Although the actual definition of agent is still debated [22, 48], there is general consensus that:

- Agents are *autonomous*, i.e., they operate without human intervention;
- Agents are *reactive*, i.e., they can react to external events;
- Agents are *pro-active* or have goal-driven behavior; and
- Agents have defined boundaries that divide internal state from the rest of the system.

Essentially, from the point of view of software engineering, an agent is an autonomous, isolated software unit with reactive and/or pro-active behavior. However, the fact that agents in simulations actually have these properties, and especially true autonomy and pro-activeness, is questioned by Drogoul et al. [12]; for discussions on multi-agent systems applied to various social network-related problems, we refer to [6, 8, 18].

When using ABM to model social networks, two kinds of relationships should be taken into account, i.e., *system-level relationships* and *network-level relationships*. The former represents all mutual dependencies, interactions and message-passing among the agents in the system, the latter are the actual relationships in the social network, so, in essence, the system-level relationships determine the system topology, while the network-level relationships determine the network topology.

The two concepts are distinct: for example, in a simulation involving a network evolution process, let nodes u and nodes v be connected in the initial state of the simulation. As a consequence, they have a network-level relationship because they are neighbors. However, if they never communicate, they are not connected by a system-level relationship. On the other hand, if at some step of the simulation u sends a message to v to sever their connection, at the end of the simulation they have no longer a network-level relationships, albeit they have a system-level relationship because of the communication. Typically, when performing social network modeling and simulation, only the network topology is analyzed. However, the network determined by the active interactions established during the simulation can also be of interest.

In ABM many different environments may be simulated, according to the actual model. However, in the case of social network simulations, the social network is the preeminent environment. Additional environmental information can be encoded in the social network, if necessary.

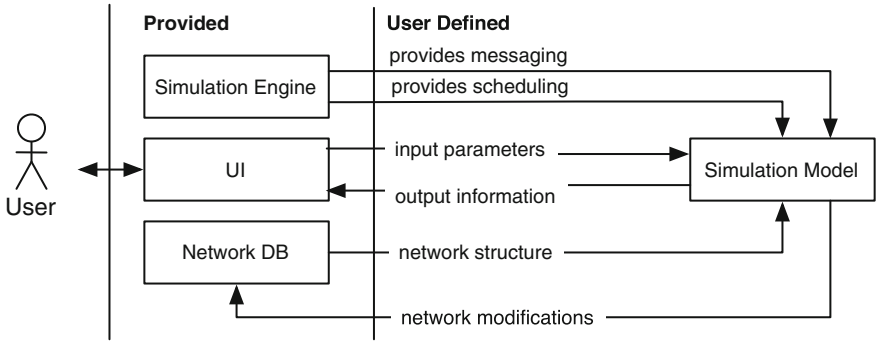


Fig. 1 Diagram representing the interactions of the four main components of a PyNetSYM simulation

2.2 Towards Agent-Based Modeling of Social Networks

In the present section we introduce the design principles of PyNetSYM,[1] a social network simulation system we propose that has the following defining goals: (1) it must support both small and large networks; (2) simulations shall be effortlessly run on remote machines; (3) it must be easy to use, even for people without a strong programming background; (4) deploying a large simulation should not be significantly harder than deploying a small one.

A PyNetSYM simulation has four main components that can be modified independently: (1) the user interface, (2) the simulation engine, (3) the simulation model and (4) the network database. The interactions between the four components are summarized in Fig. 1. The simulation model needs to be specified for each simulation and is the only part that has to be completely written by the user. Its specification is partly declarative and partly object-oriented. The user interface is responsible for taking input from the user, e.g., simulation parameters or information on the analysis to perform, and is specified declaratively. The simulation engine defines the concurrency model of the simulation, the scheduling strategies of the agents and the communication model among the agents. The network database actually holds a representation of the social network; it may be in-memory or on persistent storage, depending on the size of the simulation. Multiple network database implementations are provided and the framework can be extended with additional ones.

Large scale simulations typically require more resources than those available on a desktop-class machine and, consequently, need to be deployed on external more powerful machines or clusters. In order to simplify the operation, we designed our system so that a simulation can be entirely specified in a single file that can be easily copied or even sent by email. Considering that the simulations are frequently run on remote machines, we opted for a command line interface, because a traditional GUI becomes more complex in this scenario. An added benefit is that batch executions are

[1] https://github.com/rik0/pynetsym

also greatly simplified. We also support Read-Eval-Print Loop (REPL) interaction to dynamically interact with the simulation.

In order to allow people without a strong programming background to easily write simulations, we decided to create a *Domain-Specific Language* (DSL). A DSL is a language providing syntactic constructs over a semantic model tailored towards a specific domain (e.g., building software). The idea is that DSLs offer significant gains in productivity, because they allow the developers to write code that looks more natural with respect to the problem at hand than the code written in a General-Purpose Language (GPL) with a suitable library.

DSLs are usually categorized in *internal* and *external*: an external DSL is a new language that is compiled or interpreted (e.g., makefile), while an internal DSL is represented within the syntax of a GPL, essentially using the compiler/interpreter and runtime of the host language. Mernik et al. [33] discuss the issue thoroughly. Traditionally, DSLs were sparingly used to solve engineering problems. Nowadays, with the widespread diffusion of very expressive languages such as Python, Ruby, Haskell or Scala the general attitude towards DSLs has changed [16, 23], especially because writing internal DSLs in such languages significantly lowers the developing costs to a point that is comparable to an API based solution [27].

Consequently, we decided to develop an internal DSL embedded in Python, in order to provide an environment that is friendly enough for scientists without strong programming background, without being limiting for the others. Moreover, the internal DSL approach greatly simplifies external library usage, since all the libraries available for the host language are available for the DSL as well, and we think that accessing high quality scientific libraries and using an expressive language are both features of paramount importance. The actual description of our DSL is detailed in Sect. 4.

2.3 State of the Art of ABM

In this section, we review the most widespread ABM toolkits in more detail and compare our design decisions with theirs, where appropriate.

NetLogo focuses on the 2-dimensional grid topology where situated agents (turtles) interact with the environment and among themselves. A problem we have found with this approach is that the visualization leaks into the rest of the model. In general, we feel that the NetLogo language lacks:

- Many interesting features found in general purpose languages that may be useful to implement algorithms and data structures, e.g., (1) the lack of object types, algebraic datatypes, or even plain old record types, and (2) the need to emulate such features with agents, makes general programming awkward.
- Many useful libraries, e.g., proper social network analysis toolkits, statistical toolkits, database connectors.

In fact, using network analytic or statistical techniques is hardly uncommon inside a model, and many of these algorithms may well be as complicated to implement as the simulation itself, especially considering that NetLogo is also aimed at non-programmers.

Our approach is more similar to that of ReLogo which is a Groovy DSL inspired by NetLogo and built upon the RePast Symphony framework. Since ReLogo models are written in Groovy, they are able to call any Java library providing the required functionality. However, ReLogo is still designed with the 2D grid in mind and it does not easily inter-operate with the rest of the RePast framework [30].

Mason, RePast and Swarm follow an approach different from our own. They are mostly libraries or frameworks for general purpose languages and this is a clear advantage when interfacing with external libraries. However, all considered, those systems are relatively complex to learn because they have rich and elaborate APIs [37]. Moreover, simulations are software projects typically split in more files and rather complex deployment.

Although they are not agent-based modeling toolkits, it is worth mentioning SIENA [38] and statnet [24], two R packages created for analysis, simulation and visualization of network data. Both tools start of different premises than ABM, and are grounded in the statistical modeling of networks [42]. Statnet focuses on (1) exponential random graph models [25, 39, 40], (2) latent space models [28], and (3) latent cluster models [28], while SIENA is designed to analyze (1) longitudinal network data [41], (2) longitudinal data of networks and behaviors [43]. Considering the approach to network dynamics and the required computing resources, SIENA statistical analysis is meant for networks with a number of nodes between 10 and 1,000.

Finally, our approach shares some similarities with Pregel [32]. Pregel is a software system designed to efficiently execute graph algorithms on large relational datasets in a distributed environment. A Pregel computation takes a graph as input, and it consists in several *supersteps*, separated by global synchronization points. Within each superstep, each node executes the same user-defined function; as a result of the function execution, (1) the node state can change, and (2) the node links can change. Nodes can also communicate with messages. The main differences our approach has with Pregel are:

- The general focus: Pregel is software system to execute graph algorithms, and, although it can be used to run simulations, that is not its main purpose. Similarly, although PyNetSYM could be used to implement distributed graph algorithms, it is not optimized for the purpose.
- The approach to time and concurrency: Pregel model strictly separates the supersteps using global synchronization; on the other hand, PyNetSYM neither requires strictly distinct steps, nor mandates global synchronization. If synchronization is part of the model, it is sufficient to use an appropriate Activator agent. For more details on what an Activator is, we refer to Sect. 3.2.
- The semantics of messages: in Pregel all the messages sent at time t are received at time $t+1$. In PyNetSYM each message is available almost immediately (depending

on potential network latencies if the source and the target are not in the same process), but it is processed only when the target receives control. Although the predictability of Pregel is a desirable property, in practice, with fair schedulers, this is not much of an issue [9]. Moreover, PyNetSYM supports asynchronous and synchronous message passing semantics.

3 PyNetSYM Runtime System

The social network simulation system we propose, PyNetSYM, has an elaborate runtime system that supports the execution of simulations providing only brief specifications. The nature of these specifications is described in Sect. 4. In this section we describe (1) the components that support the simulation, (2) the general structure of the simulation execution, and (3) a metamodel to easily specify various network-based processes. We also discuss general semantic properties of the simulation engine as a concurrent system.

3.1 Simulation Engine

The central element of the runtime system is the agent, since the elements under simulation and several infrastructure components of the runtime system are implemented as agents. In the following we describe the design characteristics of PyNetSYM agents. For our purposes an agent is a bounded unit with its own thread of execution. By bounded we mean that there is a clear separation between what is *inside* the agent and what is *outside* the agent. Agents have their own state, and access to that state is mediated by the agent itself. All the communication among the agents occurs through message passing; each agent has a mailbox where the incoming messages are stored, and a unique identifier that is used to address the messages.

Agents also perceive and modify the environment. Our agents are not necessarily autonomous or goal-directed. Since they are used as a computational primitive, we need a lower-level specification that can be enriched to provide "real" agents but which does not place unnecessary overhead on the system.

The communication primitive is the **send** command. When an agent y executes the command **send**(x, "m"), (1) a message $m\{i\}$s created, (2) it is delivered in the mailbox of x, and (3) an empty placeholder is immediately returned to y as the return value of the **send** call, so that $r =$ **send**(x, "m") is a valid command. When x processes $m\{\}$, its method m is invoked and the return value is placed in r.

Send provides both the semantics of synchronous and asynchronous messaging. The semantics is that of an asynchronous message in the following cases: (1) **send** was invoked just as **send**(x, "m"), so that the return value is simply ignored, or (2) **send** was called as $r =$ **send**(x, "m"), but the caller ignores the return value r.

On the other hand, the caller y can force the retrieval of the value of r and wait until $m\{\}$ is processed. In this case, y is blocked and control is given to another agent; y will be activated again after the value of r has been supplied by x. This is essentially the semantics of a synchronous message passing. Agent y can also check without blocking whether the value of r is ready or do not block indefinitely but only up to a given timeout.

Messages with parameters can be sent with either:

$$\textbf{send}(x, \text{“m”}, p_1 = v_1, \ldots, p_n = v_n) \tag{1}$$

$$r = \textbf{send}(x, \text{“m”}, p_1 = v_1, \ldots, p_n = v_n) \tag{2}$$

In these cases the message $m\{p_1 = v_1, \ldots, p_n = v_n\}$ is delivered to x and the m method of x is invoked with actual parameters $v_1, \ldots, v_n$ passed to the formal arguments $p_1, \ldots, p_n$ of the method.

The default behavior of the agents is waiting for messages to arrive and then processing the incoming message with the appropriate handlers. However, full agent behavior (autonomous, goal-oriented, and pro-active) can be implemented either supplying a different behavior or augmenting the default with pro-activeness.

Another important design decision regarding the system semantics is whether to implement cooperative or preemptive multi-tasking. In a preemptive multitasking system, the system can (1) temporarily interrupt any active task, (2) give control to another task, and (3) give control back to the original task at a later moment. Consequently, tasks must be programmed considering that at any stage they could be interrupted and the global system state may have changed by the moment they are resumed.

In a cooperative multitasking system, the system cannot interrupt the active tasks. Each task explicitly relinquishes control in order to allow for the execution of the other tasks. Several popular languages and platform implement preemptive multi-tasking because in general purpose systems the probability and the risks of a misbehaving application consuming all the CPU time is too high. However, for a simulation oriented platform, such risks are minimal and we opted for cooperative multi-tasking because it allows a more deterministic control of complex time sequences.

As a consequence, in PyNetSYM a message handler can only voluntarily “give up” the execution for a while, either explicitly going to sleep or by requesting a blocking operation. In all other situations, when an agent starts processing a message, it continues until termination. This property is analogue to the semantics of the Actor Model [1] and simplifies formal reasoning on the system. Moreover, from the point of view of the emergent properties of the simulation it has little impact [9].

When an agent has an empty mailbox, it can choose to be removed from main memory and have its state saved on secondary storage. If the stored agent is subsequently sent a message, it is restored in main memory from the saved state. This behavior is extremely convenient considering that for most social network topologies, a small fraction of agents is responsible for the vast majority of the links. Since in most processes over networks the agents with few links are seldom activated, we

can save memory keeping them in secondary storage and do not lose much CPU time.

Another important related issue is the visibility of the network itself. A possible solution is completely distributing the knowledge of the network among the agents, so that each agent only knows its neighbors and the network must be reconstructed from their interactions. However, many network processes could not be implemented efficiently in this scenario, because several elementary operations would involve too much communication among the agents.

For example, suppose node u wants to establish a new link to another random node according to preferential attachment, i.e., with a probability proportional to the nodes degree. In this case, if there is no global view of the network, the only solution is (1) to explicitly ask every node for its degree and (2) to locally compute the probability vector. Moreover, if there is more than one node that wants to create a new link, multiple messages are sent and the computation is redundantly performed. On the other hand, if a specialized agent holds this information, the agents intending to create the links send a message to such agent and the computation is performed only once.

Consequently, we prefer to maintain a global view of the network. From the point of view of ABM, the decision is consistent with the interpretation of the network as the environment, as: (1) agents can interact with it by creating or destroying links, and (2) the agents behavior is influenced by the network in several process dependent ways.

This view is presented as a software component that we call network database (Network DB), and that provides a unified interface that agents can use to modify and query the network state. Since the Network DB is just an interface for network modifications and queries, various implementations with different trade-offs are available:

- Some implementations are RAM based, and their main advantage is to provide more efficient accesses when the network is not excessively large; others are backed with various secondary-storage based solutions, which results in slower operations, but allows for simulations on larger networks.
- Some implementations maintain the information in the memory space of the main PyNetSYM program; others use separate processes, such as SQL or NoSQL DBMSs. Setting up the runtime environment for the first kind of implementations is easier, while the latter allow to deal with larger simulations.
- Some implementations hold all the information pertaining the network structure in a single process; others split the information in multiple processes, potentially hosted on different machines. The first kind of implementations are clearly more efficient, but do not scale as easily.

All these properties can be independently chosen depending on what is desirable for the specific simulation execution. For example, the network can be stored in a MongoDB database, which is a disk-based separate-process solution. However, the database system can be run on single machine or sharded on multiple hosts. A shard is a horizontal partitioning of a database, so that rows of database tables are held

separately. The abstract nature of the Network DB leaves several possibilities open; for example, although we have not implemented it, it is possible to develop a Network DB implementation backed by Pregel [32], so that queries and network modifications are implemented in Pregel and executed on a Pregel cluster.

As a general comment, it should be said that, although PyNetSYM provides an abstract interface to the actual implementation, memory and CPU issues remain. Let us consider two different RAM based, internal and single-process implementations that are reasonable extremes regarding RAM usage. While the NetworkX library provides many useful algorithms and has a thorough support for edge and node attributes, a network of order $n = 10^6$ and size $m \sim 10 \cdot n$ occupies 4–5 GB of RAM when represented as a NetworkX graph. The reason is that NetworkX represents graphs as adjacency lists implemented using nested hash-maps. Several essential operations are fairly efficient, but substantial memory overhead is added.

On the other hand, the same network, represented as a sparse matrix occupies less than 2 MB; however, with the latter approach node and link attributes are much harder to manage, in the sense that an additional data structure is needed, which would increase memory usage considerably. Moreover, node identifiers are necessarily integers, as opposed to generic objects (including strings).

Different situations have different trade-offs and consequently we are working to make the choice of underlying implementation as transparent as possible. Moreover, depending on the actual network process, some implementations are more appropriate than others, because some operations that occur frequently in the process of choice are more efficient.

3.2 Simulation Structure

The actual simulation is divided in two distinct phases (1) setup, summarized in Fig. 2, and (2) execution, presented in Fig. 3. During the first phase (setup), the system is initialized so that it reaches the initial configuration specified by the simulation. First, various infrastructural agents (e.g., Activator, NodeManager) are created and started, so that they are ready to receive messages, and the Clock (if present) is also created, but not started. The setup phase is illustrated in Fig. 2.

Later during this phase, the Configurator agent instructs the NodeManager to (1) create the initial nodes in the network, (2) to give them instructions to establish the connections they are meant to have at t_0, and (3) to provide them with any other initial information that the simulation may require.

The NodeManager is generally responsible for (1) creating the new agent-nodes, passing them the appropriate initialization parameters and (2) monitoring them, so that their termination (exceptional or not) is managed.

We created different Configurator agents for the most frequent needs, that are (1) reading an initial network specification from disk and setting the system up accordingly, or (2) creating n node-agents of a given kind. When reading network specifications from file, we support (1) popular file formats for exchanging networks,

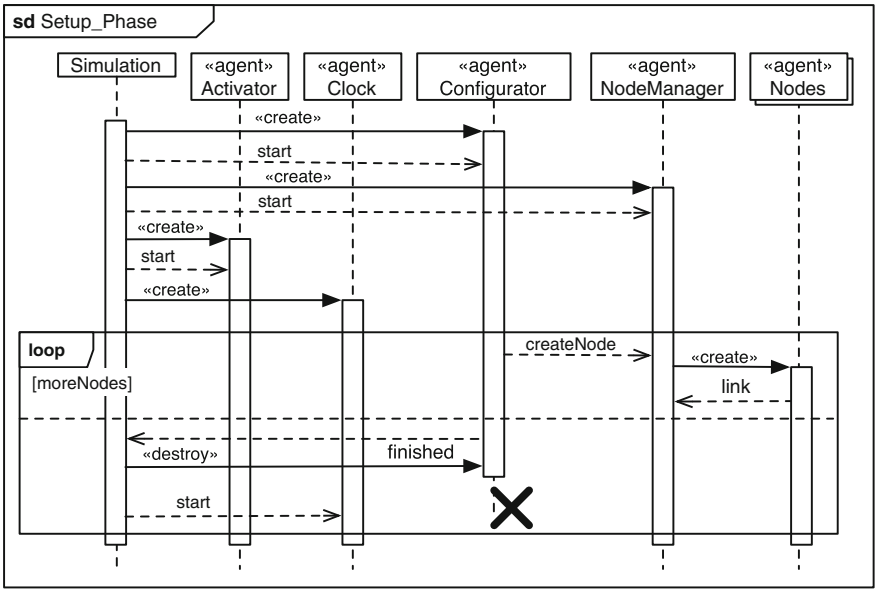


Fig. 2 Sequence diagram of the first phase (setup phase) of a PyNetSYM simulation

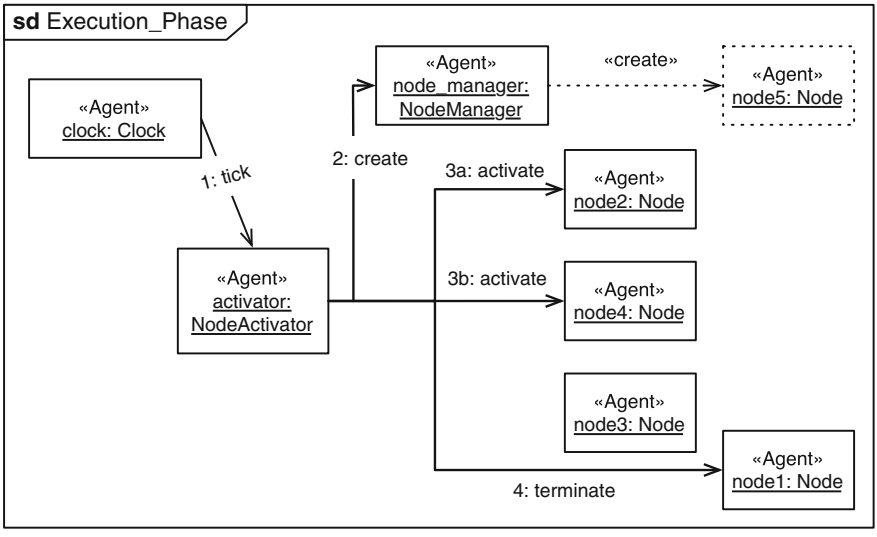


Fig. 3 Interaction diagram of the main agents in PyNetSYM during a single step of the execution phase

such as GraphML or Pajek, (2) networks saved as sparse matrices in HDF5 files, and (3) networks stored in various DBMS, such as MongoDB.

After configuring the simulation, (1) the Configurator agent terminates, and (2) if present, the Clock agent starts, marking the transition to the execution phase, or (3) the node-agents are otherwise notified that the simulation begins.

During the execution phase the node-agents perform the simulation according to their behavior. Although from a theoretical point of view such behavior can be completely autonomous and do not rely on an external time schedule, in practice most network generation models and generic processes over networks can be described in terms of a relatively simple meta-model [7, 9, 17], described below.

In the meta-model, the Clock beats the time, which is discrete. At each step, the Activator selects (1) which nodes to activate, and (2) decides whether nodes shall be destroyed, or (3) created, negotiating the eventuality with the NodeManager. The nodes execute actions after receiving the activation message. However, they can also perform actions autonomously, without waiting for activation. The nodes can also leave the simulation, require the creation of other agents, and send messages to the other nodes.

According to the meta-model, the general structure of the execution phase is presented in Fig. 3. A simulation is fully described providing the specifications of:

1. the selection process of the groups of nodes to create, activate or destroy, which is performed by the Activator agent;
2. the behavior of the nodes themselves.

Notice that the general structure does not change significantly even when introducing some agent-ness in the simulations, e.g., introducing goal-directed behavior.

4 Defining a Domain-Specific Language for Network Simulations

In Sect. 2.2 we mentioned the advantages of a DSL in terms of ease of development for both programmers and non-programmers because of the increased language expressivity. In this section we describe the general structure of our DSL, which is an internal DSL hosted by Python. We also present some examples.

Our choice of language was motivated by the following features: (1) focus in readability; (2) ease of use; (3) availability of useful libraries for scientific computations and consequently (4) widespread adoption in many scientific areas; (5) solid choice of concurrency frameworks and frameworks for distributed or GPU based computing; (6) powerful REPL implementations (7) advanced metaprogramming capabilities, which we extensively use to improve the expressiveness of the DSL. However the host language is almost an implementation detail, since other object oriented high level languages such as Ruby or Scala could have provided the same features.

Here we review some aspects of the Python language that are significantly different from static languages such as Java or C++:

(a) **class** A(B) means that class A is a subclass of class B.
(b) Methods are defined with the **def** keyword. For clarity, we differentiate among methods and message handlers; the latter are introduced with the **handler** keyword, even though this keyword is not part of Python's syntax.
(c) The explicitly specified formal parameter "self" is the object the method is invoked on.
(d) Methods can be invoked with named parameters.
(e) Set ($\{a, b\}$) and hash-map ({k : "v", . . .}) literals are available.
(f) Everything defined directly in the class body becomes a class attribute. A callable defined in the class body is a method.
(g) Classes are first class objects: when called, are factories.
(h) Classes can be defined inside other classes; in this case, they are "attributes" of the enclosing class. As a consequence, an inner class can be invoked as a method of the enclosing class and returns the appropriate instance object.
(i) The **send**(x, "m") command discussed in Sect. 3 becomes a method defined in the Agent class and, as such, is invoked with self.send(x, "m").

As described in Sect. 3.2, our simulations have two distinct logical elements:

1. The essentially imperative/object oriented description of the agents behavior (e.g., the nodes and the Activator agent).
2. The mostly declarative description of the simulation itself and of the simulation options specification.

Regarding the imperative/object oriented part, it suffices to say that the nodes and the Activator agent are implemented as subclasses of Agent, which means they are essentially Python objects with agent semantics provided by PyNetSYM and discussed in Sect. 3.1. Their behavior is thus specified using the host language (Python) for maximum expressivity.

The other logical element defining a model is the Simulation class. A Simulation is not an agent and is essentially the executable specification of the simulation that collates together all the other elements. A Simulation object has a `run` method that is called to execute the simulation. When `run` is called, both (1) command line arguments and (2) actual parameters directly passed into `run` are taken into account, i.e., it processes the "simulation options" field and creates a command line parser that can parse command line options according to the specification.

The specification is a list of allowed options, each with its name, default value and the function to convert the string value as specified in the command line to the proper value. For example, the following specification would let the simulation accept two options, *option1* and *option2*; the first option has default value 1.0 and is of type float, the second has 1 as the default value and is an integer:

```
command_line_options = (
    ('--option1', {'default': 0.0', 'type': float}),
    ('--option2', {'default': 1, 'type': int}))
```

Moreover, a help message is automatically constructed from the options, and is displayed if the simulation is run with the "`--help`" option. Such message lists the various options and, if a documentation string was provided in the specification, also their meaning.

When a simulation is executed, it always instantiates some objects and agents that fill pre-determined roles, such as the network database, the Configurator agent, the Activator agent, or the Clock agent.

Additionally, the behavior of simulations can be declaratively customized. If the body of the Simulation subclass has an attribute in the form `role_type`, it is used as the factory for the object in question in place of the default implementation. For example, a different Activator can be requested with the `activator_type` option.

A subset of the simulation options is passed to the constructors of objects and agents instantiated by Simulation. The subset is determined in the following order: (1) inspecting the "options" attribute in the object class definition, (2) using an attribute named role_options defined in the Simulation class, and (3) performing introspection on the names of the parameters of the object/agent constructor. Introspection is a programming language feature that allows programs to analyze types and function/method signatures at runtime. Additionally, the Simulation class has an `additional_agents` attribute, where additional "roles" can be defined.

If the configuration is wrong or incomplete (e.g., because of a missing or wrongly typed parameter), the simulation fails as early as possible, ideally at "compile time" (i.e., when classes are evaluated).

4.1 SIR Model

In this section we present the implementation of the Susceptible-Infected-Recovered (SIR) model in our DSL. The SIR model was originally proposed to study the outbreak of contagious illnesses in a closed population over time [26] and was subsequently adapted as a social network process [36]. For our purposes, the SIR model is the network adapted variant. In this form, each node has three possible states: susceptible (S), infected (I) and recovered (R), hence the SIR name. The system starts with a given ratio r of infected patients, and each infected patient can recover with probability γ. Moreover, each infected patient infects each of its neighbors with probability β.

We fit the SIR model to our meta-model so that, at each step, the Activator agent only activates infected nodes (Fig. 4, lines 6–7). Susceptible and recovered nodes do not need to take action. Consequently, when a node is infected, it sends a message to the activator to inform it that it is infected and, similarly, when it recovers, it sends a message indicating its recovery. When an infected node is activated, it tries to spread the disease among its neighbors by sending them messages. Afterwards, it may recover with a given probability (Fig. 5).

In the implementation with PyNetSYM the simulation has three initial parameters: the infection rate β, the recovery rate γ and the initial ratio of infected nodes r. The declaration of the simulation options is given in lines 26–29 of Fig. 6.

```
1: class SIR_Activator(pynetsym.Activator):
2:     handler infected(self, node):
3:         self.infected_nodes = self.infected_nodes.add(node)
4:     handler not_infected(self, node):
5:         self.infected_nodes = self.infected_nodes.remove(node)
6:     def nodes_to_activate(self):
7:         return self.infected_nodes
```

Fig. 4 SIR model activator implementation

```
 8: class SIR_Node(pynetsym.Node):
 9:     handler initialize(self, state):
10:         self.state = state
11:         if state == "I" :
12:             self.send(Activator.name, "infected", node=self.id)
13:     handler infect(self):
14:         if self.state == "S" :
15:             self.state = "I"
16:             self.send(Activator.name, "infected", node=self.id)
17:     handler activate(self):
18:         if state == "I" :
19:             for node in self.neighbors():
20:                 if random.random() < self.infection_rate :
21:                     self.send(node, "infect")
22:             if random.random() < self.recovery_rate :
23:                 self.state = "R"
24:                 self.send(Activator.name, "not_infected", node=self.id)
```

Fig. 5 SIR model node implementation

An additional starting parameter is the shape of the network. The initial configuration of the network is read from a file, hence we use NXGraphConfigurator (line 32 of Fig. 6), that (1) reads files in any format supported by NetworkX, and (2) it creates the appropriate nodes along with their connections. The type of the nodes and the simulation options they require are specified with the node_type and node_attribute of the Configurator respectively. Finally, initialize_nodes is a special method of the Configurator agent which is called after the nodes have been created to provide them with additional setup. In this case, it randomly selects which nodes start infected.

In Figs. 4 and 5 the implementation of the SIR nodes and of the SIR Activator are shown. Essentially, their behavior is that described when discussing how to fit the SIR model to the meta-model.

In Fig. 7 we plot the evolution of the fraction of susceptible, infected and recovered nodes as functions of the elapsed number of steps as given by the PyNetSYM simulations of the SIR model. The starting network is an Erdős-Rényi graph $G(10^5, 120)$ [15], i.e., a random graph where each of the 10^5 nodes has 120 random edges. The 1 % of the nodes starts as infected ($r = 0.01$) and the parameters here are infection rate $\beta = 1.0$ and recovery rate $\gamma = 0.4$.

```
25: class SIR_Simulation(pynetsym.Simulation):
26:     simulation_options= {
27:         ("--infection-rate", {"default" : 1., "type" : float}),
28:         ("--recovery-rate", {"default" : .4, "type" : float}),
29:         ("--initial-infected-rate", {"default" : .01, "type" : float})}
30:     activator_type = SIR_Activator
31:     activator_options = {}
32:     class configurator_type(pynetsym.NXGraphConfigurator):
33:         node_type = SIR_Node
34:         node_options= {
35:             "infection_rate",
36:             "recovery_rate" }
37:         def initialize_nodes(self):
38:             infected_nodes = random.sample(
39:                 len(self.node_identifiers) * self.initial_infected_rate)
40:             for node in self.node_identifiers:
41:                 if node in infected_nodes :
42:                     self.send(node, "initialize", state="I")
43:                 else:
44:                     self.send(node, "initialize", state="S")
```

Fig. 6 SIR model simulation specification

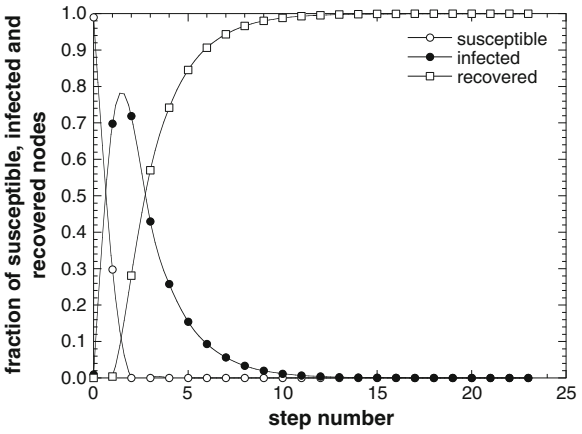


Fig. 7 Time evolution of the fraction of susceptible, infected and recovered nodes as functions of the number of steps according to the PyNetSYM agent-based simulation of the SIR model. The starting network is a $G(10^5, 120)$ network, the starting parameters are $\beta = 1.0, \gamma = 0.4, r = 0.01$. The *solid lines* are guides for the eye

4.2 Barabàsi-Albert Model

In Fig. 8 we show the code for a simulation implementing the Barabàsi-Albert (BA) model [3]. Line numbers reported in this section refer to Fig. 8 unless differently specified. The BA model starts with n_0 nodes and no edges. At each step a new

```
1:  class BA_Node(pynetsym.Node):
2:      handler activate(self):
3:          targets = set()
4:          while len(targets) < self.starting_edges :
5:              target = self.graph.preferential_attachment()
6:              if target not in targets :
7:                  targets.add(target)
8:                  self.link_to(target)
9:  class BA_Activator(pynetsym.Activator):
10:     options = {"starting_edges"}
11:     handler tick(self):
12:         answ = self.send(NodeManager.name
13:             "create_node", cls=BA_Node,
14:             parameters= [self.starting_edges])
15:         self.send(answ.get(), "activate")
16: class BA_Simulation(pynetsym.Simulation):
17:     activator_type = BA_Activator
18:     simulation_options= {
19:         (network_size, {"default": 100, "type": int}),
20:         (starting_edges, {"default": 5, "type": int})}
21:     class configurator_type(BasicConfigurator):
22:         node_type = BA_Node
23:         node_options = {"starting_edges"}
```

Fig. 8 BA model simulation implementation

node with m random links is added. The m links are directed towards nodes with a probability proportional to their degree, this strategy is called *preferential attachment*.

In Lines 16–23 the simulation configuration is specified. Line 17 specifies that the BA_Activator (defined in lines 9–15) is the activator to be used. Classes can be stored in variables like any other object.

Lines 18–20 specify the command line options, their default values and the type of the parameters. When values for some options are not specified in the command line, the default value is used.

The BasicConfigurator (Lines 21–23) reads the value n_0 of the simulation option "`--network-size`" and requests to the NodeManager the creation of n_0 nodes of type specified by the `node_type` attribute.

In Lines 1–8 and 9–15 there are the specifications for the nodes and the activator respectively. The nodes are specified providing a handler for the *activate*{} message. Moreover, in the Node class we defined:

- A method link_to, which requests the creation of a link with a given node, and whose default implementation simply sends a $link\_request\{source\}$ message to the target node, as shown in Lines 3–4 of Fig. 9.
- A handler for $link\_request\{source\}$ messages. The default implementation provided in Lines 5–7 of Fig. 9 always accepts the link and informs the agent managing the network to create the link. In a model where nodes can decide whether to accept a connection, such default handler would be overridden with the desired behavior.
- A conceptually similar method/handler pair is provided to sever links.

```
1: class Node(Agent):
2:     ...
3:     def link_to(self, target):
4:         return self.send(target, "link_request", source=self.id)
5:     handler link_request(self, source):
6:         self.send(GraphAgent.name, "add_link", source=source, target=self.id)
7:         return True
8:     ...
```

Fig. 9 Implementation of some node methods and handlers

The Activator accepts a *tick*{} message from the clock. In Line 11 the Activator sends the NodeManager: (1) the class of the agent to create in the "cls" parameter and (2) the arguments to pass to their constructor with the "parameters" parameter. This is a synchronous call, because the Activator blocks waiting for the identifier of the newly created agent. Then, it immediately sends to the new agent an *activate*{} message (line 15).

5 PyNetSYM Concurrency Approach

In Sect. 3 we defined the runtime model of our simulation framework. Essentially, we assumed that agents do have their own thread of control, but did not specify how concurrency was implemented. The traditional approach in agent-based frameworks is to implement concurrency mainly with threads, and to provide cooperative multitasking as an option [4]. Even when cooperative multitasking is available, agent platforms are typically built around threading features provided by the hosting platform [5, 19].

Instead, we decided to use gevent,[2] a networking and concurrency framework that uses coroutines to implement cooperative tasks, called "greenlets" in the gevent lingo. Coroutines are a generalization of subroutines allowing multiple entry points for suspending and resuming execution at certain locations [11].

A regular procedure has only one entry point, i.e., the beginning of the procedure, reached after the procedure invocation, and potentially multiple exit points, i.e., the return statements. On the other hand, coroutines can be exited and re-entered, so it is possible to start a coroutine f_1, exit at a given point and start executing a coroutine f_2 and successively re-entering f_1 at the point where the execution was stopped, actually resuming the previous computation. As a consequence, it is possible to use coroutines to implement cooperative multitasking, as long as a task scheduler is provided. In our case, Gevent also provides the greenlets scheduler.

Frameworks such as gevent are popular for writing programs with very high concurrency, since greenlets: (1) are less expensive to create and to destroy; and (2) do not use memory structures in kernel space. In fact, greenlets live entirely in the

[2] http://www.gevent.org

Table 1 Comparison between Greenlet and thread efficiency: the main agent spawns a new agent and sends it a message with the numeric value of the number of agents to spawn, then it waits for a message on its own queue

# items	Thread execution (s)	Greenlet execution (s)
1×10^2	0.022	0.002
5×10^2	0.110	0.009
1×10^3	0.224	0.018
1×10^4	2.168	0.180
1×10^5	21.85	1.796
1×10^6	223.3	18.24

When the other agent receives a message with value n, each secondary agent: (1) spawns a new agent and sends it a message with the value $n - 1$ if $n \neq 0$; (2) on the other hand, if $n = 0$, it means that all the agents have been created and sends a message to the main thread. Notice that the agents do not exist all at the same time: once they have received and sent a message, they terminate

user-space, thus context switches between different greenlets are inexpensive as well and do not involve system calls.

In Table 1 we report execution times of a simple benchmark performed with a different number of threads/greenlets. It is easy to see how the greenlet based solutions are roughly ten times faster than the corresponding thread-based ones. Further comparisons between greenlet and thread performances can be found in [20].

The better performance of greenlets is particularly relevant in our case, since in ABM it is required to potentially support millions of concurrent agents, and, since greenlets use no kernel resources, their number is limited only by the amount of available RAM. Moreover, experience suggests that a modern operating system can only support a few thousand threads.

Although, thread pools offer a viable, albeit more complex solution, cooperative multitasking as provided by greenlets gives a finer grained control over scheduling and concurrency in general, because the moments when context switches occur become more predictable, as they cannot be externally interrupted. Moreover, the lack of preemption relieves from the necessity to protect critical sections, i.e., portions of codes that modifies data structures shared among multiple tasks.

6 Conclusion

In this chapter, we have presented PyNetSYM, a novel language and runtime for network specific simulations. PyNetSYM is built for the generative approach [13] to science typical of Agent-Based Modeling (ABM). We believe there is a strong need for tools that are both: (1) easy to use (especially for people with little programming background but with a significant expertise in other disciplines, such as social sciences) and (2) able to tackle large scale simulations, using remote high-performance machines and potentially distributing the computation on multiple servers. Therefore, while our primary goal is maintaining our toolkit simple and easy to use, efficiency

is our second priority, since nowadays there is a wide interest on networks of huge size.

Specifically, we created PyNetSYM: (1) to easily support both small and huge networks, using either in-memory and on-disk network representations, and (2) to be as easy to be used both on personal machines or on remote servers.

We designed PyNetSYM so that all the entities, both the infrastructural ones and those under simulation, are agents: defining the network simulation is essentially a matter of specifying (1) the behavior of the nodes and (2) a few additional simulation parameters (e.g., storage strategy and user-customizable options).

Given the merits of Domain-Specific Languages (DSLs) in general, and specifically the ones concerning how to make development easier both for programmers and non programmers alike, we decided to create a DSL to drive PyNetSYM simulations, so that it is possible to write programs that are machine-executable, high-level formulations of the problem to solve. Specifically, our DSL is an internal DSL over Python.

As PyNetSYM provides the simulation engine, the simulation can be written in a simple file using our DSL. Thus, PyNetSYM models are very easy to deploy (copying around a single file is sufficient) and can also be effortlessly shared between researchers. Moreover, PyNetSYM models can also be written and executed interactively from a Read-Eval-Print Loop (REPL), a feature that is extremely useful when exploring the results of a simulation, because it makes possible to interactively and dynamically perform analysis and visualize data.

We also implemented some classic processes over networks (generation models, infection/information diffusion processes) with PyNetSYM and found that the resulting programs are readable and fluent; in fact, they are very similar to a pseudo-code formulation of the models, even though they are efficiently executable. At the present moment we are using PyNetSYM to simulate the behavior of users in a novel fully distributed social networking platform, in order to understand the condition under which the information propagates to the intended recipients. Although, the study is at a rather early stage, we find the PyNetSYM modeling features quite satisfactory.

Our results show that our approach is successful in providing a friendly and easy to use environment to perform agent-based simulations over social networks. Agent-based simulation is a powerful conceptual modeling tool for social network simulations and with the present work we contribute a natural and expressive way to specify social network simulations using a DSL.

Acknowledgments Stimulating discussions with Prof. A. Poggi are gratefully acknowledged.

References

1. Agha G (1986) Actors: a model of concurrent computation in distributed systems. MIT Press, Cambridge
2. Axelrod R (2007) Simulation in social sciences. In: Rennard J (ed) Handbook of research on nature-inspired computing for economics and management. IGI Global, Hershey, pp 90–100

3. Barabási AL, Albert R (1999) Emergence of scaling in random networks. Science 286:509–512
4. Bellifemine F, Caire G, Poggi A, Rimassa G (2008) Jade: a software framework for developing multi-agent applications. Lessons learned. Inf Softw Technol 50(1–2):10–21
5. Bergenti F, Franchi E, Poggi A (2010) Using HDS for realizing multiagent applications. In: Proceedings of the third international workshop on languages, methodologies and development tools for multi-agent systems (LADS'10), Lyon, France, pp 62–68. http://CEUR-WS.org
6. Bergenti F, Franchi E, Poggi A (2011) Agent-based social networks for enterprise collaboration. In: Proceedings of IEEE 20th international workshops on enabling technologies: infrastructure for collaborative enterprises. IEEE, Paris, France, pp 25–28
7. Bergenti F, Franchi E, Poggi A (2011) Selected models for agent-based simulation of social networks. In: Kazakov D, Tsoulas G (eds) Proceedings of the 3rd symposium on social networks and multiagent systems (SNAMAS '11). Society for the Study of Artificial Intelligence and the Simulation of Behaviour, York, UK, pp 27–32
8. Bergenti F, Franchi E, Poggi A (2012) Enhancing social networks with agent and semantic web technologies. In: Brüggemann S, D'Amato C (eds) Collaboration and the semantic web: social networks, knowledge networks, and knowledge resources. IGI Global, pp 83–100
9. Bergenti F, Franchi E, Poggi A (2013) Agent-based interpretation of classic network models. Comput Math Organ Theory 1–23. doi:10.1007/s10588-012-9150-x
10. Boyd DM, Ellison NB (2008) Social network sites: definition, history, and scholarship. J Comput Mediated Commun 13(1):210–230
11. De Moura AL, Ierusalimschy R (2009) Revisiting coroutines. ACM Trans Program Lang Syst 31(2):6:1–6:31
12. Drogoul A, Vanbergue D, Meurisse T (2003) Multi-agent based simulation: where are the agents? In: Sichman J, Bousquet F, Davidsson P (eds) Multi-agent-based simulation II. Springer, Berlin, pp 43–49
13. Epstein JM (1999) Agent-based computational models and generative social science. Complexity 4(5):41–60
14. Epstein JM, Axtell R (1996) Growing artificial societies: social science from the bottom up. Complex Adaptive Systems Series. MIT Press, Cambridge
15. Erdős P, Rényi A (1959) On random graphs. Publicationes Mathematicae 6(26):290–297
16. Fowler M (2010) Domain-specific languages. Addison-Wesley signature series (Fowler). Addison-Wesley Professional, Reading
17. Franchi E (2012) Towards agent-based models for synthetic social network generation. In: Putnik GD, Cruz-Cunha MM (eds) Virtual and networked organizations, emergent technologies and tools, communications in computer and information science, vol 248. Springer, Berlin, pp 18–27
18. Franchi E, Poggi A (2011) Multi-agent systems and social networks. In: Cruz-Cunha M, Putnik GD, Lopes N, Gonçalves P, Miranda E (eds) Business social networking: organizational, managerial, and technological dimensions. IGI Global, Hershey
19. Franchi E, Poggi A, Tomaiuolo M (2013) Developing pervasive and adaptive applications with MAADE. J Comput Sci Appl 1(1):5–13. doi:10.12691/jcsa-1-1-2
20. Friborg RM, Bjørndalen JM, Vinter B (2009) Three unique implementations of processes for PyCSP. In: Welch PH, Roebbers H, Broenink JF, Barnes FRM, Ritson CG, Sampson AT, Stiles GS, Vinter B (eds) Communicating process architectures 2009—WoTUG-32, concurrent systems engineering, vol 67. IOS Press, pp 277–293
21. Gardner M (1970) The fantastic combinations of john conway's new solitaire game "life". Sci Am 223:120–123
22. Genesereth MR, Ketchpel SP (1994) Software agents. Commun ACM 37(7):48–53
23. Ghosh D (2010) DSLs in action. Manning Publications
24. Handcock MS, Hunter DR, Butts CT, Goodreau SM, Morris M (2008) Statnet: software tools for the representation, visualization, analysis and simulation of network data. J Stat Softw 24(1):1–11
25. Hunter DR, Handcock MS, Butts CT, Goodreau SM, Morris M (2008) Ergm: a package to fit, simulate and diagnose exponential-family models for networks. J Stat Softw 24(3):1–29

26. Kermack WO, McKendrick AG (1927) A contribution to the mathematical theory of epidemics. Proc Roy Soc Lond A 115:700–721
27. Kosar T, Martínez López PE, Barrientos PA, Mernik M (2008) A preliminary study on various implementation approaches of domain-specific language. Inf Softw Technol 50(5):390–405
28. Krivitsky PN, Handcock MS (2008) Fitting position latent cluster models for social networks with latentnet. J Stat Softw 24(5)
29. Luke S, Cioffi-Revilla C, Panait L, Sullivan K, Balan G (2005) Mason: a multiagent simulation environment. Simulation 81(7):517–527
30. Lytinen SL, Railsback SF (2012) The evolution of agent-based simulation platforms: a review of NetLogo 5.0 and ReLogo. In: Proceedings of the fourth international symposium on agent-based modeling and simulation, Vienna, Austria
31. Macal CM, North MJ (2010) Tutorial on agent-based modelling and simulation. J Simul 4(3):151–162
32. Malewicz G, Austern MH, Bik AJC, Dehnert JC, Horn I, Leiser N, Czajkowski G (2010) Pregel: a system for large-scale graph processing. In: Proceedings of the 2010 ACM SIGMOD international conference on management of data. ACM, Indianapolis, Indiana, pp 135–146
33. Mernik M, Heering J, Sloane AM (2005) When and how to develop domain-specific languages. ACM Comput Surveys (CSUR) 37(4):316–344
34. Minar N, Burkhart R, Langton C, Askenazi M (1996) The swarm simulation system: a toolkit for building multi-agent simulations. In: Technical report. Santa Fe Institute, Santa Fe
35. North MJ, Howe TR, Collier NT, Vos JR (2007) A declarative model assembly infrastructure for verification and validation. In: Advancing social simulation: the first world congress. Springer, Japan, pp 129–140
36. Pastor-Satorras R, Vespignani A (2001) Epidemic spreading in scale-free networks. Phys Rev Lett 86:3200–3203
37. Railsback SF, Lytinen SL, Jackson SK (2006) Agent-based simulation platforms: review and development recommendations. Simulation 82(9):609–623
38. Ripley RM, Snijders TAB, Preciado P (2011) Manual for RSiena. University of Oxford, Department of Statistics, Nuffield College
39. Robins GL, Pattison P, Kalish Y, Lusher D (2007) An introduction to exponential random graph (p*) models for social networks. Soc Netw 29:173–191
40. Robins GL, Snijders TAB, Wang P, Handcock M, Pattison P (2007) Recent developments in exponential random graph (p*) models for social networks. Soc Netw 29:192–215
41. Snijders TAB (2005) Models for longitudinal network data. In: Carrington P, Scott J, Wasserman S (eds) Models and methods in social network analysis, Chap 11. Cambridge University Press, New York, pp 215–247
42. Snijders TAB (2011) Statistical models for social networks. Annu Rev Sociol 37(1):131–153
43. Steglich CEG, Snijders TAB, Pearson M (2010) Dynamic networks and behavior: separating selection from influence. Sociol Method 40:329–393. doi:10.1111/j.1467-9531.2010.01225.x
44. Stroud D (2008) Social networking: an age-neutral commodity—Social networking becomes a mature web application. J Direct Data Digital Mark Pract 9(3):278–292
45. Tisue S, Wilensky U (2004) NetLogo: a simple environment for modeling complexity. In: International conference on complex systems, Boston, MA, USA, pp 16–21
46. Van Dyke Parunak H, Savit R, Riolo R (1998) Agent-based modeling vs. equation-based modeling: a case study and users' guide. In: Sichman J, Conte R, Gilbert N (eds) Multi-agent systems and agent-based simulation. Lecture notes in computer science, vol 1534. Springer, Berlin, pp 277–283
47. Von Neumann J, Burks AW (1966) Theory of self-reproducing automata. University of Illinois Press
48. Wooldridge M, Jennings NR (1995) Intelligent agents: theory and practice. Knowl Eng Rev 10(02):115–152

Early Stage Conversation Catalysts on Entertainment-Based Web Forums

James Lanagan, Nikolai Anokhin and Julien Velcin

Abstract In this chapter we examine the interest around a number of television series broadcast on a weekly basis. We show that through analysis of initial conversation between fans or users of dedicated web forums we can provide a description of the greatest period of interest (or *peak*). We then focus our attention on this peak with an ultimate goal of characterising episodes as a function of the actions and qualities of the people that take part in early conversation about this episode. We find that early interaction trends have strong similarities with the overall conversation patterns, and contain the majority of information provided by influential members of the community. This observation has important implications for the rapid generation of meta-data which may be used during later broadcast and re-runs for description and valuation of episodes.

1 Introduction

Recent years have seen a massive increase in research into real-time information retrieval, providing strong evidence of the growing importance of temporal information. It is essential to remember however that the nature of the information being shared in real-time is necessarily less in-depth than that of, say, web forums which are not as time-sensitive. Within a web forum it is easy to browse past discussions and posts of users, even those from many hours or days prior to the current moment, and respond. As a result users are able to write more and consider topics at a higher level than any one current issue.

J. Lanagan (✉) · N. Anokhin
Technicolor, 1, avenue de Belle Fontaine—CS 17616, 35576 Cesson-Sévigné Cedex, France
e-mail: james.lanagan@technicolor.com

J. Velcin
ERIC Lab, 5 av. Pierre Mendès-France, 69676 Bron Cedex, France

F. Can et al. (eds.), *State of the Art Applications of Social Network Analysis*,
Lecture Notes in Social Networks, DOI: 10.1007/978-3-319-05912-9_5,

This type of interaction is not yet possible on real-time websites due to the amount of information being generated. The method of delivery is more akin to a reservoir where information is available for a fixed period of time before being washed away. Although real-time services such as Twitter provide a means of gauging instant reaction to events such as political rallies and live sports competitions [10, 27], the enduring nature of web forums and other online resources means that conversations can take place over longer periods of time at a more sustained volume. While this information is not truly created from real-time sharing, the style of content being discussed can affect both the reaction and interest surrounding it.

The viewing habits of television and internet users are also changing, meaning that there is an ever-increasing desire for content on demand. While real-time information can help to guide viewing habits and choices at the time of broadcast, analysis of sources such as forums can help in providing deeper more targeted recommendations at a later date. Analysing dedicated forums allows us to understand fans' reactions to specific episodes or series, and provide recommendations based on these insights rather than a purely content-based approach.

It has always been important for advertisers and content providers to know audience sizes and opinions when calculating market value. Gensch and Shaman [17] used time series of raw viewing figures to predicting the ratings and subsequent earning of television series. Their approach could be improved upon by incorporating the reactions of viewers as well as these viewing figures: advertising costs could be adjusted during later broadcasts of episodes that have been received particularly well or badly.

While it is interesting to look at the reactions of the audience as a whole, focussing on those users who are most influential or informed provides the greatest chance of generating useful meta-data to aid the above goal of revenue increase. This has implications when deciding the fate of early-stage television ventures also. It is therefore important that this content is not lost due to shortened observational time windows.

In this chapter we examine the interest around a number of television series that are broadcast on a weekly basis. We perform rapid analysis of conversation generated by forum users' posts for a given episode, and analyse comments posted so as to provide a description of the principal point of interest (the *peak*). We then focus our attention on this peak and evaluate the behaviour of users within it compared to during the entire conversation life-cycle. The ultimate goal is the characterisation of episodes as a function of the actions and qualities of the people that take part in this discussion so as to use this information during later broadcast, video on demand, or similar services.

The major contributions of this chapter are two-fold:

- We show that it is possible to cluster the trend within conversation threads, and use the clustered conversation peak *period of interest* to characterise the evolution of the thread early in its life-cycle.
- We show that it is possible to identify those user within the forum who act as "conversation catalysts" for subsequent analysis. It is also shown that limiting this

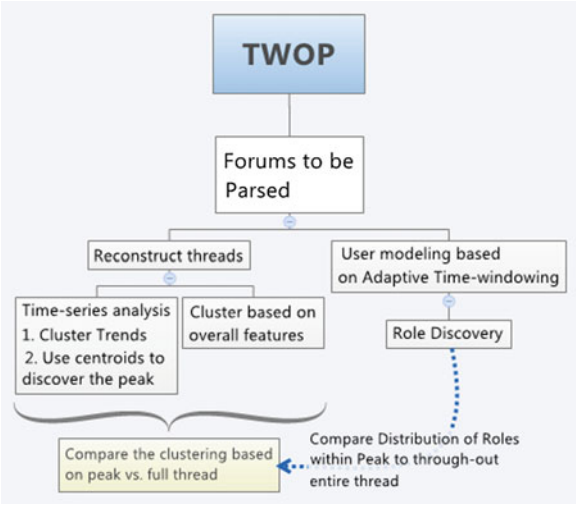


Fig. 1 The workflow of our approach

analysis to the aforementioned "period of interest" retains the majority of these catalysts' posts.

By considering the contributions of users to a forum within an adaptive sliding time window, we are able to perform unsupervised ensemble clustering on the temporal representations of users to assign a principal (most representative) role. This extends past work [3] that used measures adapted from the citation analysis domain to assign roles. We show that our measures help to differentiate more clearly why certain users are ranked highly within the social network.

In the experiments that follow we shall examine how much information can be learned about an episode given just the first week of activity after broadcast. We shall further restrict ourselves to the time during the initial "peak" of discussion. We are able to show that restricting analysis to the conversation peak still allows for analysis of the conversations between the most prominent and important users within the forum. This procedure is illustrated in Fig. 1.

Note that here we would like to show that it is possible in the context of forum discussions to restrict analysis of conversation to early interactions about a show rather than waiting for the thread to die. This restriction allows for increased scalability (through reduction of the volume of content being analysed) as well as speed since, as shown in Sect. 4, the periods of interest last just a few days. While this may seem slow in the context of real-time analysis, it is adequate in the context of TV and Video-on-demand (VOD) recommendations where users do not require the same level of immediacy. It is also important here to recall that our ultimate goal lies closer to the realm of advertising revenue maximisation through increased understanding of how content was received by the audience.

In the next section we detail related work before discussing the collection, preparation and composition of our corpus in Sect. 3. Section 4 provides a background on time-series, as well as an explanation of method to discover the period of interested mentioned above in Sect. 4.1.1. Details of the new adapted measures that we have created to characterise our forum users are listed in Sect. 4.2. The experimental procedure is outlined in Sect. 5. Finally in Sect. 6 we draw some conclusions from our experiments and elaborate on future research directions.

2 Related Work

The related work to our own stretch across two fields; those of social network analysis and temporal analysis. Both of these fields have been researched extensively and so we will not provide an exhaustive list but instead some of the more directly related research concerning entertainment content and its meta-data, much of which comes from web forums and dynamics around user interaction.

Work on predicting the final box-office earnings of movies as well as future ratings from content features and meta-data has been performed in the past [22, 31]. These approaches use a variety of data ranging from IMDb ratings to the text content of both professional and amateur critical reviews. All of these approaches centre on cinematic releases that are inherently one-time/unique. Our work is specific to meta-data (at present forum posts) created about television series built on episodic content allowing us to examine the activity around several similar pieces of content.

Temporal information plays an important role in the understanding of forum and online social dynamics. Activity on websites and forums is bursty by nature, and having identified these bursts it is possible to weight posts as a function of the general interest at a specific time-point [33]. Yang and Leskovec [40] observe the temporal clustering patterns around internet memes; interest is implicitly qualified by the shape of these patterns.

By looking at the evolution of users within a forum, Kan et al. [23] look to classify the usage patterns of users. Their approach however disregards explicit temporal information such as the time a user performs an action or creates a forum post. Rowe et al. [35] use content and structure specific features to predict how threads within web forums will grow over time. They look to the readability, polarity and focus of users to predict whether posts within a forum will receive replies. Will this approach is useful in observing the growth potential of the conversation, the temporal aspect of conversation is again ignored.

The shape of associated financial temporal information is also used by Lavrenko et al. [28] to categorise text and build language models about stocks. Analysis of the time series of public opinion is performed by O'Connor et al. [30] so as to predict the outcome of political polls. This work requires time series or long duration however, due both to the volume of data and its subsequent fluctuations.

Web forum and news group interactions have been well researched for many decades, and more recently these ideas have been migrated to the field of social

media network analysis [14]. The structural properties of threads within forums, both their reconstruction and growth have helped in understanding how forum users interact [37]. The usefulness or value of specific posts within a forum has been also studied [8, 26], allowing for moderation as well as information filtering.

In the literature there are two dominating approaches to detecting the social roles of users in a social network. The first approach consists of extracting interesting qualitative and quantitative features from the structure and content of discussions, and then performing a comparative analysis of these features' values across users [5, 39]. The main difficulty of this approach is that in order to apply supervised learning, one first has to have a labelled training data set. Assigning labels, however, is difficult due to subjectivity notions of influence. This may be overcome by using unsupervised learning techniques [40].

Dynamic block-models have been used in the past to observe the changing roles that users can have [15]. The dynamics of relations/links between actors in the social network are used to attribute the roles. In contrast, we perform our role assignment by looking at the behaviour of users in general throughout our observation period, and not simply by the roles of each user's neighbours.Unlike the multiple-role approach however [38], we attribute a single role per time-windows but this role can change between time-windows.

The second approach models influence as an epidemic, or the spread of a disease in a population represented as a network of users [1, 11]. However, unlike presence or absence of illness, the fact of having an opinion is vague and subjective. Moreover, the traditional "epidemic" approach only takes into consideration the structure of the user network, while in the context of influence it makes sense to consider other factors as well.

There is little work on using citation analysis approaches with the context of web forums [3, 18]. Examining the reactions to the posts of users across a forum allows improved understanding of what makes a person central to the network or forum community. Garfield observed 15 separate reasons for the citation of articles within a work [16]; Brooks notes that it may in fact be far more complex than this [7]. The reasoning remains strongly linked to the reasoning behind online conversations; people search for social consensus or validation of their own opinions [2].

Yu et al. [41] use semantic similarity of content to improve TV personalization. They define similar TV contents to be those with similar semantic information, e.g., plot, background, genre, etc. and propose a hybrid approach for TV content similarity, which combines both the vector space model and category hierarchy model. It measures TV content similarity from the semantic level than the physical level. This approach differs from our own as the purpose here is not for recommendation of content, but more the discovery of similarities within the reaction of the audience itself. This has the advantage of being less content-specific, and focussed instead towards possibilities for revenue maximisation.

In this chapter we focus on the growth and clustering of temporal information around entertainment content. While we want to perform early clustering and characterisation based on a peak period of interest, we also wish to show that limiting analysis/information to this peak does not significantly impact future research on the

textual content itself. We therefore also study the distribution of user types within the conversation threads. Before analyse begins however, we detail the collection and preparation of the forum content.

3 Corpus Creation

Our corpus was created by crawling 6 years of forum posts from TWOP[1] crawled in February 2012. The website is dedicated to entertainment discussion, and the 7 television series we have analysed feature in the "Top Shows" categories (containing 8–10 television series) on the site[2]: American Idol, Breaking Bad, Dexter, Grey's Anatomy, House, Mad Men, and The (American) Office. These series were picked for both their critical success and on-site audience in an attempt to ensuring a fan-base and number of comments of sufficiently large size.

TWOP contains dedicated sub-forums each focussed on a single television series (Fig. 2), allowing us to tag each thread in our corpus with a single series of interest. Many topics may be discussed on the forums—future ideas or speculation; actors; theories—but for the work that follows we are interested solely in those threads that discuss particular episodes. Details of how we recognised those threads discussing single episodes are given below. From this point on, when we refer to episodes we mean a single broadcast of a series for which a thread exists within our corpus (Table 1). In order to have a significant number of posts, we ignore the initial season (if available) allowing the audience and fan base to build. Doing so reduces our corpus to 278,037 posts by 20,465 unique authors (This number is different from that in Table 1 as some authors are active in more than one sub-forum).

We focus our analysis of role distribution within threads on the years 2007–2008. This is done for practical reasons of scale. Limiting our analysis to these 2 years results in a loss of 1 of our 7 shows,[3] but messages are still spread across 220 threads. As a result of pre-processing 1,959 forum users were retained for analysis (see Sect. 5.2).

Corpus Preparation

After crawling the forum, it was necessary to reconstruct the threads of discussion: TWOP uses a 'quote' mechanism meaning that the quoted text of a message appears before the text of a reply. We used a series of regular expressions, as well as Levenshtein distance (95 % overlap) to detect the parent-child relationship within the past 20 in-thread messages. This threshold was chosen empirically as it provided us with

[1] http://forums.televisionwithoutpity.com/

[2] Other than American Idol, the 6 series are a spread of entertainment genres from drama, suspense, and comedy. American Idol is unique amongst the collection as it is a reality TV competition meaning that both the number of episodes and the level of interest in any particular episode is somewhat different from a weekly series.

[3] Breaking Bad was first mentioned on the site in 2009.

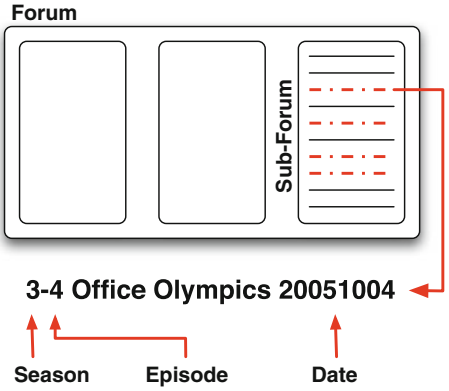


Fig. 2 Each sub-forum contains threads that are focussed on a single episode of a particular series

Table 1 Breakdown of corpus by sub-forum

Forum	Seasons (analysed)	Threads (episodes)	Posts	Authors
American idol	7	323	93,307	8,793
Breaking bad	3	25	8,406	579
Dexter	6	60	11,609	1,390
Grey's anatomy	8	147	47,673	4,921
House	8	141	47,767	3,953
Mad men	4	37	32,178	2,080
The office	8	145	37,097	3,668
Totals	44	878	278,037	25,384

This is the corpus of episode-specific threads after removal of Season 1

high retrieval. We manually checked the efficacy of the proposed thresholds across 10 % of the corpus and found a 100 % retrieval rate for all quoted parent texts.

While the sub-forums are focussed on particular series, the topics of conversation within any single thread can be varied. There is however a convention used on the forum as a whole to signify discussion about a particular episode that allows us to filter and retrieve (see Fig. 2) just these threads and their posts. The title of any thread discussing a single episode is of this form allowing us to analyse just these focussed conversations.

Having retrieved a conversation thread for each episode in a series, we are then able to analyse the temporal posting patterns of users on a per-episode basis. The nature of the series that we are studying is such that a new episode is broadcast each week. This has lead us to the following question: if we examine just the first week after a episode is broadcast, how well can we characterise all conversation about the episode? Is it necessary to study the whole week (or longer) when we see so much of the activity focussed in the short period after broadcast?

4 Evolution of Conversation Threads

We look at the time series created by the hourly interactions of users on the sub-forums of interest for the first week post-broadcast. Every forum has its own life-cycle: with respect to the forums we are studying, there is a regular burst of activity the night of a new episode's airing and for some time after. This burst of activity is of particular interest to us. As we have stated, early identification of the reaction to shows could help with pricing of advertising during later broadcasts. To better understand this peak we must also have an understanding of the participants. In this section we shall therefore first explain how we will identify the peak activity within thread time series, and next detail how we can classify the users within the forums (thus threads) by their behavioural roles.

4.1 Temporal Analysis

In order to apply time series analysis it is necessary to regularise and smooth the time series as there are many hours during which there are no interactions. The natural circadian rhythm of forum users creates an irregular time series: the decomposition of time series (as described below) requires that there be a uniformly distributed fixed sampling interval, and therefore that the time series to be analysed, be regular. We create a mapping from our irregular time series to a uniformly distributed hourly time series, aligning the posting times to the nearest hour. We also insert zeros for all hours (such as early morning) where no posts have been made, thereby creating a regular time series to be smoothed.

Trends within Time Series

Any (additive) time series, x_t, may be decomposed into its constituent parts that describe the overall series in terms of particular features [9]. In general, a systematic change in a time series that does not appear to be periodic is known as a trend, m_t, and is modelled by the *trend component*. The time series that we observe within our sub-forums are indeed additive, allowing us to perform time series decomposition and study the underlying TREND within our episode conversation threads [24]. Removing the seasonal effect (a repeating pattern within the time series) of the circadian rhythm allows us to observe the actual trend in interest around a particular episode.

The trend is a CENTRED MOVING-AVERAGE of width frequency, F, and centred on the point of interest, t. In our time series of hourly posting activity this frequency corresponds to a 24 h day:

$$\hat{\mathrm{m}}_t = \frac{1}{2F} x_{t-\frac{1}{2}F} + \frac{1}{F} \times \sum_{j=-\frac{1}{2}F-1}^{\frac{1}{2}F-1} x_{t-j} + \frac{1}{2F} x_{t+\frac{1}{2}F} \tag{1}$$

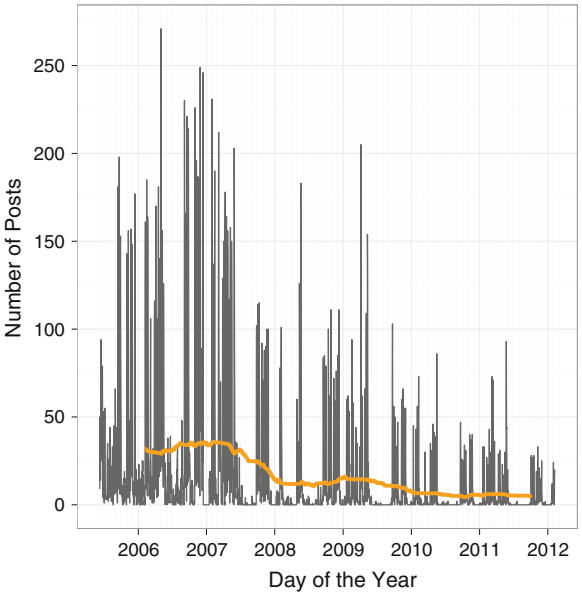


Fig. 3 Overall posting behaviour for 'House'. The trend ($F = 365$ days) is shown in *orange*

As shown in the above formula $\frac{F}{2}$ values are required before t, and so the hourly trends do not have observations before $t = 12$. We begin our time series on the broadcast date of episodes: the shows of interest are all broadcast in the evening and so are there are more than $\frac{F}{2}$ hours before the majority of posts. This is an important consideration however for shows which are broadcast at an earlier time. It may be necessary to begin time series on the day before broadcast in some cases.

Although the trend creates a smoothed or filtered version of the original series with many of the spikes of activity removed, it also provides us with a far better understanding of the continued interest around an episode. We use thread trend information as the starting point of our analysis.

Clustering Similar Trends

Having identified the trend for each thread/episode, we would like to verify that these trends may be used to anticipate and understand the longer-term reaction by the community to an episode. Figure 3 shows the overall posting behaviour of users on the 'House' sub-forum. We can see that the overall interest in the series displays a decreasing trend ($F = 365$ days), however the activity around any one episode can defy this general trend.

One might assume a somewhat natural clustering presented by simply grouping on seasons (September–May), thus grouping highly-discussed episodes from later seasons with those that provoked far less discourse simply because of their proximity in broadcast date. This would however be naïve: even though the general volume of comments in later seasons may has decreased, the speed and intensity with which

users/fans greet a great episode does not. For this reason we choose to examine the shape of the trend (interest) curve for discussion about each episode, and cluster episodes as a function of this curve. In order to study this we must first normalise the volume, thereby ensuring that the time series is indeed a function of the interest rather than simply volume. For this we perform min-max normalisation[4] prior to decomposing and reducing the time series into its trend information.

Since our time series have been regularised, they are all of equal length $T = [t_0, t_n]$ and can be considered as vectors. Unlike in the case of "meme" detection [40], our time series begin and end within fixed time periods ($t_0 = broadcastdate$ to $t_n = t_0 + 7$), and so translation should not in fact be invariant. Translation invariance implies that small-scale conversation delayed by several days should be considered in a similar light to that happening in volume soon after broadcast. By observation we see that the conversation that takes place just after broadcast is far more involved/reactionary than later. It is not of the same type as memes where the meme has simply been discovered/shared by a new audience: here the audience is already focussed on the topic. For these reasons we believe K-MEANS CLUSTERING is an appropriate means of clustering our 8-day time series trend data. The choice of k is detailed below in Sect. 4.3.

4.1.1 Conversation Peaks

The clustering centroids obtained for the Grey's Anatomy series are shown in Fig. 4. Here we can see that the norm is to post soon after the broadcast of an episode, though not always to the same degree. We would like to further examine this burst of peak activity and in doing so characterise the conversation surrounding each episode. We can see in Fig. 4 that it would not be wise to simply take the first x hours after an episode is broadcast and choose this as a cut-off point for "peak interest" since the peak itself can be of dramatically different shape (cf. Cluster 1 vs. Cluster 2).

More formally we observe $T = [t_0, t_n]$ the 8 days surrounding the broadcast of a show, and look to define a period of interest, $i = [c_s, c_e]$ contained within T such that a) it is possible to predict the interest that a particular show has received as a function of its assigned cluster centroid and b) to do this as early as possible. We choose to use the 2nd *half-life* of the maximum value (a reduction to 25 % of the trend maximum value) as a cut-off when defining i. Taking the first time point before, c_s, and after, c_e, the peak maximum fulfils this requirement and defines i. Our choice of threshold is based on both empirical analysis and past work [40]. Unlike Yang and Leskovec however, the nature of our content and audience means that the period of greatest interest/activity falls far faster after it peaks.

As stated, the shape of this peak can vary, but we found that the majority of peaks are between 24 and 72 h duration (minimum peak duration was 18 h, maximum 84 h; median of 41 h). The limits of i are always aligned to the nearest hour as this is a property of the time series created in Sect. 4.1.

[4] Min–Max normalisation creates values in the range [0, 1] using the formula: $x = \frac{x - min(x)}{max(x) - min(x)}$.

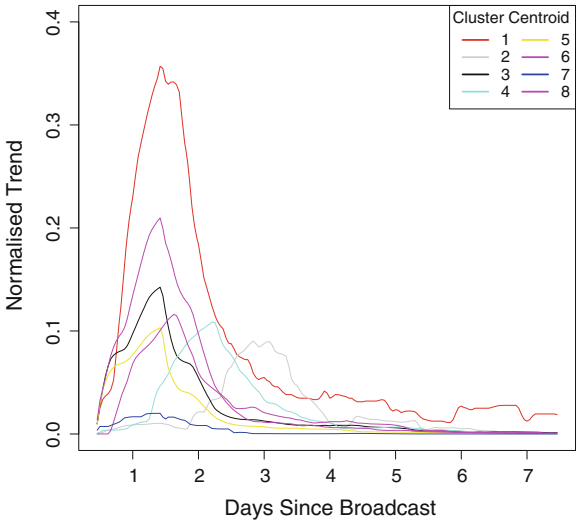


Fig. 4 Clusters obtained on the Grey's anatomy trend time-series

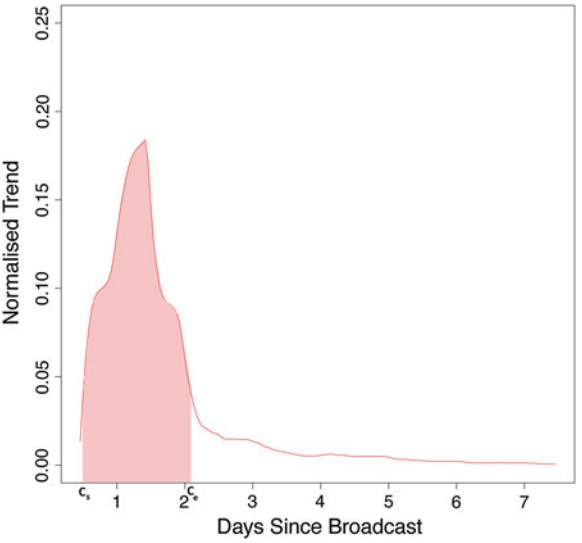


Fig. 5 Clustering on the peak period (shaded) of the time series

Every thread in a particular cluster, c, is now limited or 'cut-off' using the cluster centroid. i.e. every thread in c is shortened to contain only those posts that were created between the hours c_s, the start of the centroid's peak, and c_e, the end of the centroid's peak (Fig. 5). We will examine the features of these new peak-limited threads and their ability to characterise and predict the features of the *entire* thread.

By this we refer to the full thread of discussion present in our corpus and not just the first week.

The ultimate aim of discovering a limited time-series i is to investigate and understand the reaction to an episode. We may in future like to perform qualitative research on the conversation and so it is important that limiting our study to i does not mean losing a large quantity of important/insightful content from important key community members. It is therefore necessary to define what we mean by "important" community members.

4.2 Bottom-Up Role Identification

While understanding of conversation growth may be possible from just early interest, it is important to gain an understanding of the users who are showing this interest. We look to the interactions of a forum as they change over time, and how these interactions place certain users into certain behavioural roles. At present the most important of these is that of the influential users. These influential users, or "conversation catalysts" are able to spark discussion and their comments may be seen as a rich source of metadata to aid in classification and understanding of the reactions to particular episodes.

We will use the approaches put forward in the citation analysis literature to provide a measure of interaction and importance to our forum users. This differs from purely graph-based measures as we intend to take into account only those interactions that have been judged sufficiently interesting. We then use these distinctions in importance and interaction style to help in role detection. We adapt two well-known citation metrics to define "sufficiently interesting" posts. These are Hirsch's *h-index* [20].

> An author has an **h-index** of h if h of his total contributions have received at least h citations each (the H-CORE).

and its successor *g-index* [12]:

> An author has a **g-index** g if g is the highest rank such that the top g contributions have, together, at least g^2 replies. This also means that the top $g + 1$ messages have less than $(g + 1)^2$ replies.

The proposed methodology is an extension of the work of Anokhin et al. [3] who identified social roles based on the evolution of user features over time using a fixed sliding time window. We extend their work by looking at the directed graph of user interactions over activity-based varying time-windows.

Defining Behavioural Characteristics
Adapting the measures of Anoknin et al. and applying them to our corpus helps us understand the differing roles played by our communities' users.

Node (Neighbours') g-(In/Out) Index This feature evaluates how active neighbours of the node are. Node out-g-index is the feature that characterizes users who often replies to users who reply to other users. High Node g-In-Index shows that a user is often replied to by users who get a lot of replies themselves. This is indicative of

being well known/respected by other well-known users. in[out]-g-index is calculated as the highest number g of predecessors[successors] such that sum of their weighted in[out]-degrees is g^2 or more.

Catalytic Power The Catalytic Power of a message k reflects the amount of reaction from other users caused by the message: in the context of forum discussions, we use the number of direct replies, c_k, to estimate this power. The catalytic power of a user A is consequently defined as the sum of powers from the H-CORE of all messages sent by A.

Weighted In/Out Degree The weighted out[in]-degree of a node A at moment t is defined as the sum of all communications from[to] A to[from] all neighbours B, $(A \rightarrow B)$, in its neighbourhood $N(A)$.

Activity The activity of a forum user clearly affects the way the user is perceived by other members of the forum. To measure a user's activity we use the number of messages posted by the user. It has been noted [1] that high activity alone does not guarantee an important role in the forum.

Cross-Thread Entropy Let us consider a user who posted i_d messages across all d threads. Let $n = \sum_i i$, then focus of a user is defined as:

$$F = \sum_d -\frac{i_d}{n} \log \frac{i_d}{n}.$$

This measure helps to distinguish between users who contribute to many threads across the forum from users who focus on a single thread. We also calculated the topic entropy but found this to be too discriminative as most users in our sample are only interested in a single topic/show.

For every user we calculated a time series of daily feature vectors. Each feature vector contains seven values: a user's in/out degree, g-in/out index, catalytic power, activity, and cross-thread entropy.

Adaptive Time Windowing

Although analysis of users' features obtained from the entire data sets reveals general patterns of interactions between users, it is also important to consider how values of those features evolve over time. With TWOP we observe weekly peaks of activity on the broadcast dates of new episodes of each series, and almost no activity in summer when none of the six TV shows are broadcast. There exist weeks when the activity of users was very small, meaning even the influential users are "underestimated" as they simply don't have enough users to communicate with.

To cope with these effects we use variable-width time windows, rather than constant ones. The width of each window is adjusted in order to contain at least $\mu_{msg} - \sigma_{msg}$ messages as the time windows before it, where (μ_{msg}) and (σ_{msg}) are the mean and standard deviation of the number of messages within all time windows up to this point. Figure 6 shows the adaptive time windows computed over the years 2007–2008.

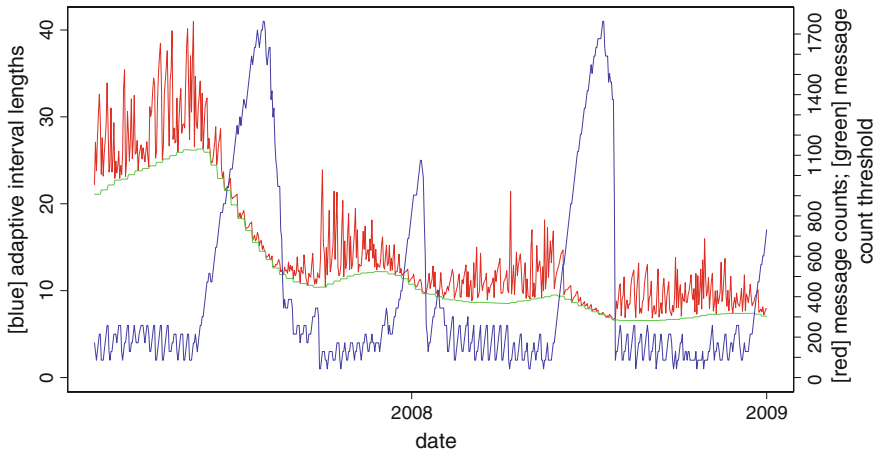


Fig. 6 Adaptive time windows: *blue*—lengths of time windows in days, *red*—numbers of messages in time windows, *green*—$(\mu_{msg} - \sigma_{msg})$

It can be seen that in the periods of extremely high activity the time windows contracted to the length of 1–3 days, while low user activity is characterised by very long time windows (about 6 weeks in July 2007).

Every user is now represented by a 7 dimensional time-series of daily feature vector observations. The activity of each user is far from uniform, and consequently most users are likely not to play the same social role all the time: in the current work we shall assign a user to their most representative role. Although in future we plan to consider temporal evolution patterns of the proposed features to identify roles, we currently assume that each social role is characterised by a feature vector with certain values of its components. In order to find these values, we use unsupervised learning to cluster our users' time points.

Ensemble clustering

The particular feature of the given data is that it contains a large number of data items. Not all conventional clustering algorithms can deal with such large amounts of data. ENSEMBLE CLUSTERING is a possible approach that allows us to overcome this problem [4]. This approach is robust to fluctuations within the data, and also provides good stability and accuracy for clustering partitions. It is also a faster clustering technique which is necessary for our role detection step as there is for more data than in the case of our episode threads.

We follow the process set out by Ayad et al. [4] in which s sub-samples of the entire data-set are taken with replacement, and then each s is clustered into l clusters. A subsequent meta-clustering is performed in which these sample clusterings are combined to provide a final stable clustering of k clusters. This finally meta-clustering can be performed using any clustering technique: we choose to use hierarchical clustering in this final phase as was proposed in the original publication.

4.3 Validating the Number of Clusters

We cluster our time-series of episode-centric thread posts using k-means clustering. It is necessary however to use ensemble clustering for our user time-series for the reasons previously stated. K-means clustering is a particular case of ensemble clustering when the initial l and final k number of clusters are the same. Both these algorithms are thus unsupervised processes, but it is necessary to provide the final number of clusters, k, to be created. Out of several existing techniques we chose three indices that are orthogonal by construction and so provide a more reliable estimation of the correct k clusters. To find the optimal value of k, each index is computed for $k = 1, 2, \ldots, n_{max}$ clusters.

Note that the validation indices are computed using k-means. The reason is that it is a non-trivial task to adapt ensemble clustering to validation indices. Previous work has shown however that in the case of the time-point clustering, each cluster corresponds to a certain *behaviour* of a user and these behaviours can still be captured using k-means [3].

The HARTIGAN INDEX [19] for n items into k partitions is defined by the relative measure of the reduction in square error, when the number of clusters increases from k to $k+1$. The main disadvantage of applying Hartigan's index to our data is that the number of items in the data set is so large that the scaling factor $(n-k-1)$ does not make any difference. The KL-INDEX [25] is an attempt to overcome this difficulty; the number of data never appears in the expression for KL-Index explicitly.

The STABILITY INDEX [6] uses a different idea: instead of validating the number of clusters using within-cluster distance, it assumes that for a good k the clustering has to remain roughly the same even for a subset of data.

5 Experimental Analysis

Our experiments are a two stage process: firstly we identify the peaks within the episodic threads and compare clusters of peaks with entire threads based on a number of quantitative measures. We then examine if the use of just these peaks for future community analysis results in a major loss of information with regards our catalysts.

5.1 Clustering Episodes

We would like to cluster the episodes of each of our series of interest based on the features that we obtain by examining the initial burst or peak of activity. We then compare this clustering with one based on the full-length (sometimes lasting over a year) threads. Our goal is to enable an analysis based on the first few days after broadcast that is equivalent to one performed many days or weeks later.

In order to characterise our conversation threads it is necessary to define a set of features that best describe them. We choose to use a set of features that model both

the scale of conversation as well as the involvement or interaction (e.g. quoting and replying). We follow previous work [13, 26] that has looked to define features of both the authors and posts within a forum thread:

- NUMBER OF AUTHORS: The number of distinct users that have taken part.
- NUMBER OF POSTS: The total number of posts.
- NUMBER OF ORIGINALS: The number of posts that are not replies to another post.
- NUMBER OF REPLIES: The number of posts that are in reply to a previous post.
- AVG. POST LENGTH: The average character count of posts.

The first two features describe the amount of general interest surrounding the thread. More users may be seen as an indication of wide-spread interest. The latter three features are indicative of the intensity of discussion within the thread. The more evenly balanced the ratio of original posts to replies, the more likely it is that a conversation of greater depth is taking place. In this work we remain focussed on the quantitative features of the threads rather than qualitative and so disregard any further in-depth analysis of the content of threads themselves. We do provide some further thoughts on possible areas of research in Sect. 6.

We now perform a second round of clustering but this time on the feature vectors of the peak-limited and entire duration threads. Again we use Hartigan's Index to decide on the number of clusters to use, as well as 100 iterations of k-means. Table 2 show the number of clusters used for each clustering.

5.1.1 Comparison of Clustering Results

In order to compare the clustering of each of our feature vector representations, we use the well-known ADJUSTED RAND INDEX (ARI) [21]. This is a measure of accuracy or agreement between two clustering solutions in the range $[-1, 1]$. The general RAND INDEX [32] is the number of pairs of elements, a, that are assigned to the same cluster in both partitions; plus the number of pairs of elements, b, that are assigned to different clusters in both partitioning; over all possible pairs of elements:

$$\text{ARI} = \frac{\text{Index} - \text{Expected Index}}{\text{Max Index} - \text{Expected Index}} \tag{2}$$

Here the value of ARI is zero when the index equals its expected value (under the generalized hypergeometric distribution assumption for randomness) [36]. It is important to note that the number of clusters chosen for our two datasets is not always equal.The reasons for this different number of clusters could be that the range of values that particular features take is smaller within the peak than across the time series as a whole. The inverse may also be true where threads limited to only the peak activity exhibit active conversation, but as the thread grows longer (more time passes) the effect of this initial burst of conversation is dampened.

We can see that in general there is reasonable agreement in the clustering produced on both the entire and peak periods of a thread (Table 2). We believe this, along with

Table 2 Rand index for the agreement of clustering for entire and peak-limited thread feature vectors

Forum	Clusters (entire)	Clusters (peak)	Number	rand index	% posts in peak
American tdol	4	4	52,003	0.00	49.69
Breaking bad	4	9	300	0.19	60.97
Dexter	4	5	1,170	0.30	48.45
Grey's anatomy	4	4	10,731	0.27	40.82
House	4	4	9,870	0.27	52.31
Mad men	4	4	666	0.04	54.94
The office	4	5	10,440	0.33	32.08

the high percentage of posts found within the peak, shows strong justification for concentrating analysis on the peak period since this is a good characterisation of the final thread.

The worst performance is seen on the *American Idol* and *Mad Men* sub-forums. Mad Men has a far shorter seasons than any of the others (with the exception of *Breaking Bad*) and so there are less threads against which to perform a clustering (see Table 1). This also means that there is less variation within the features of peak-period threads, again impacting the clustering. American Idol is the inverse having very long seasons and large volumes of commentary. This commentary is provided by large numbers of users (mainly new/insignificant) and so the features used to classify the complete threads are more varied. We believe there is a lot of mixing between clusters resulting in a low ARI. Further analysis is warranted into the affects of both media and post content on the amount of conversation produced, as well as the degree to which the length of a season affects clustering.

5.2 Behaviours Discovered

As stated in Sect. 3 we focus our analysis of role distribution within threads on the years 2007–2008. For every user we calculated a time series of 730 feature vectors, one for each day. Each feature vector contains seven values: a user's in/out degree, g-in/out index, catalytic power, activity, and cross-thread entropy.

The feature vectors with all-zero components were excluded as they represent moments when users were not part of the community.[5] We also choose to filter out all users who have not written at least 10 messages during our chosen timespan as we do not have sufficient information to infer a behavioural role to them. As a result 139,474 feature vectors belonging to 1,959 forum users were retained.

Ensemble clustering was applied to the feature vectors of forum users with the following parameters: samples $s = 40$; re-sampling probability $p = 0.5$; clusters for

[5] They may have posted a new (non-reply) message, but have received no replies to this message. Hence all values apart from activity are zero.

Table 3 Summary of cluster centroids for feature vector from ensemble clustering

Cluster	In degree	Out degree	In g-Index	Out g-Index	Catalytic power	Cross-thread entropy	Activity
1	0.070	0.075	2.26e−4	0.052	1.59e−01	1.98e−01	0.300
2	0.010	0.221	0.000	0.000	7.97e−05	5.98e−03	0.205
3	0.291	0.048	0.322	0.000	2.34e−01	7.52e−02	0.241
4	0.428	0.429	0.447	0.445	3.65e−01	1.80e−01	0.445
5	0.242	0.305	0.277	0.335	1.93e−01	5.52e−03	0.287
6	0.008	0.000	0.000	0.000	0.000	2.43e−16	0.171
7	0.012	0.271	0.000	0.308	2.74e−05	8.07e−03	0.231

samples $l = 35$; Sample clustering algorithm: k-means with 1,000 re-initialisations; final clusters $k = 7$. Table 3 presents the centroids of the clusters created through the ensemble clustering process. It should be noted that these centroids are not exact representations of the clusters (since hierarchical clustering does not produce the same elliptical clusters as k-means) but provide strong indication of the points within each cluster.

These cluster centroids help to classify user behaviour, although it is not immediately obvious as to what behaviour *every* cluster corresponds. We can however see a number of obvious divisions that have been captured. Cluster 3 for example contains all observations where a user participated in several threads (hence the relatively low entropy) and received significant amount of replies (high in-degree) from prominent people (high in-g-index), although not replying much (low out-degree), making them something of a catalyst as evidenced by the cluster's high catalytic power.

Cluster 2 on the other hand shows evidence of times when users are unable to generate any conversation (low catalytic power & in/out-g-index) despite having replied to many comments in a wide variety of threads (high out-degree & entropy). Cluster 6 represents non-active users (lowest activity). Such users do not get many replies; do not reply much; and consequently are not likely to be known in the community.

Two clusters presented in Table 3 are of special interest for us. Cluster 4 includes feature vectors with high values of all the features; members of this cluster can be seen as potential catalysts in several topics. Cluster 5 also has high-connected characteristics (in/out(-g)-indexes) but are unable to generate interaction (low catalytic power). This may be due to their far less focussed behaviour as witnessed by low entropy.

5.3 Participation of Thread Lifespan

We are interested to know if disregarding all messages that happen outside of the peak would result in significant loss of insightful information. We characterise this

Table 4 Median percentage of role contributions found within the peak

Forum	Episodes	Role 1	Role 2	Role 3	Role 4	Role 5	Role 6	Role 7
American Idol	73	0.46	0.52	0.63	0.80	0.95	0.98	0.00
Dexter	24	0.52	0.79	1.00	0.00	0.00	0.00	0.00
Grey's Anatomy	40	0.31	0.45	0.70	0.91	1.00	0.00	0.00
House	41	0.37	0.57	0.71	0.94	1.00	0.00	0.00
Mad Men	13	0.35	0.47	0.74	0.94	0.99	0.00	0.00
The Office	36	0.33	0.38	0.64	0.79	0.97	0.00	0.00

by saying that the retention of information produced by important users is critical. Table 4 shows the percentage of posts that are retained per role if only the peak is considered. It can be see that while at least half of all messages produced by lesser users is lost (Roles 2, 6, & 7), practically all posts created by important or nearly important users (Roles 4 & 5) is retained. Role 7 is a vary sparsely populated role which explains the lack of posts within the peak. We see that Dexter again suffers from the misclassification of the peak.

This result shows that it is possible to use just the peak period to gather meta-data about the episode—semantics, descriptions, named entities—without fear of losing large amounts of content contributed by influential users.

6 Conclusion and Future Work

The focus of the work presented here has been on the quantitative aspects of online forum discussions. We have shown a method for focusing on the initial buzz around new episodes of television series, and validated that this focus significantly biases neither the overall structure of conversation threads nor the presence of its most important contributors. This focus does however present many interesting future challenges with respect to qualitative analysis.

We would like to study the correlations between fan-based/critic episode ratings and the temporal patterns/clusters found using the approaches described in this chapter. It would also be interesting to observe the correlation between critical ratings and the participation of the users we have identified as important using our proposed features.

As stated previously, the role that a user plays may change over time [15]. Although our approach helps to identify important users despite inactivity for a period of time, we see that there is important work to be continued on the dynamic evolution/transference from one role to another. Deciding on an appropriate size time window is key to discovering the role of a user at any time-point; we will examine further appropriate time-intervals for the graph snapshots [29], as well as [34] time-respecting sub-graphs.

The differences in the number of clusters produced for Breaking Bad illustrate one limitation of the approach we have taken. Many posts appear to happen outside of the peak period for threads in this sub-forum. Manual observation of the threads shows that there is one episode that received over twice the comments of all others on the forum. A means of recognising bursts of activity within threads is required to alleviate this problem [33].

It would be of interest to study different types of content so as to assertain the generalisability of the our approach. The particularity of the content we have studied is that it has weekly nature meaning that most conversation happens soon after broadcast leading to obvious peak interest. This may not be true for sports/live content or movies. The application of event detection approaches similar to those of Lanagan and Smeaton [27] can alleviate this problem: here only the content around significant events could be studied. Once this event detection is performed, the approach outline here may serve to improve understanding and characterisation of those events.

In this paper we have seen that by looking at the evolution over just a few days it is possible to predict the overall shape and trend within conversations of longer duration. Clustering of these peak periods of conversation allows us to focus our attention on the initial buzz and excitement generated by an episode. We have also shown that by focussing on just this peak, we do not lose significant amount of information that has been produced by the most central and dedicated members of the community. This discovery alone has important implications for those interested in the quick identification and classification of content for such low-level purposes as meta-data extraction or indeed high level ones such as advertising valuation.

Acknowledgments The authors would like to thank the reviewers for their valuable comments on the content of this paper.

References

1. Agarwal N, Liu H, Tang L, Yu PS (2008) Identifying the influential bloggers in a community. WSDM '08. Palo Alto, CA, USA, pp 207–218
2. Agosti M, Ferro N (2006) Annotations on digital contents. Int J Digit Libr 6:124–138
3. Anokhin N, Lanagan J, Velcin J (2011) Social citation: finding roles in social networks. An analysis of TV-series web forums. In: COMMPER 2012, p 49
4. Ayad HG, Kamel MS (2005) Cluster-based cumulative ensembles. MCS'05. Seaside, CA, pp 236–245
5. Bakshy E, Hofman JM, Mason WA, Watts DJ (2011) Identifying 'influencers' on twitter. In: WSDM'11, Hong Kong, China
6. Ben-Hur A, Elisseeff A, Guyon I (2002) A stability based method for discovering structure in clustered data. In: Pacific symposium on biocomputing, pp 6–17
7. Brooks TA (1986) Evidence of complex citer motivations. JASIS 37(1):34–36
8. Chai K, Hayati P, Potdar V, Wu C, Talevski A (2010) Assessing post usage for measuring the quality of forum posts. In: DEST'10, IEEE, pp 233–238
9. Cowpertwait PSP, Metcalfe AV (2009) Introductory time series with R. Springer, Berlin
10. Diakopoulos NA, Shamma DA (2010) Characterizing debate performance via aggregated twitter sentiment. CHI '10: proceedings of the 28th international conference on human factors in computing systems. ACM, Atlanta, Georgia, USA, pp 1195–1198

11. Domingos P, Richardson M (2001) Mining the network value of customers. In: KDD '01, San Francisco, CA, pp 57–66
12. Egghe L (2006) Theory and practise of the g-index. Scientometrics 69(1):131–152
13. Fisher D, Smith M, Welser HT (2006) You are who you talk to: detecting roles in usenet newsgroups. In: HICSS'06, vol 3, IEEE, p 59b
14. Forestier M, Velcin J, Zighed DA (2011) Extracting social networks to understand interaction. ASONAM'11. IEEE Computer Society, Kaohsiung, Taiwan, pp 213–219
15. Fu W, Song L, Xing EP (2009) Dynamic mixed membership blockmodel for evolving networks. In ICML'09, ICM, pp 329–336
16. Garfield E (1997) Concept of citation indexing: a unique and innovative tool for navigating the research literature
17. Gensch D, Shaman P (1980) Models of competitive television ratings. J Market Res 17:307–315
18. Gómez V, Kaltenbrunner A, López V (2008) Statistical analysis of the social network and discussion threads in slashdot. WWW '08, Beijing, China, pp 645–654
19. Hartigan JA (1975) Clustering algorithms. Wiley, London
20. Hirsch JE (2005) An index to quantify an individual's scientific research output. PNAS 102(46):16569–16572
21. Hubert L, Arabie P (1985) Comparing partitions. J Classif 2(1):193–218
22. Joshi M, Das D, Gimpel K, Smith NA (2010) Movie reviews and revenues: an experiment in text regression. HLT '10. Los Angeles, CA, pp 293–296
23. Kan A, Chan J, Hayes C, Hogan B, Bailey J, Leckie C (2012), A time decoupling approach for studying forum dynamics. Arxiv, preprint arXiv:1201.2277
24. Kendall M, Stuart A (1983) The advanced theory of statistics, vol 3. Griffin, London
25. Krzanowski WJ, Lai YT (1988) A criterion for determining the number of groups in a data set using sum-of-squares clustering. Biometrics 44(1):23–34
26. Lanagan J, Smeaton AF (2009) Query independent measures of annotation and annotator impact. ESAIR '09. ACM, Barcelona, Spain, pp 35–38
27. Lanagan J, Smeaton AF (2011) Using twitter to detect and tag important events in live sports. In: ICWSM '11, AAAI
28. Lavrenko V, Schmill M, Lawrie D, Ogilvie P, Jensen D, Allan J (2000) Mining of concurrent text and time series. In: KDD-2000, workshop on text mining, ACM, Boston, MA
29. Lijffijt J, Papapetrou P, Puolamäki K (2012) Size matters: finding the most informative set of window lengths. ECML-PKDD'12. Bristol, England, pp 451–466
30. O'Connor B, Balasubramanyan R, Routledge BR, Smith NA (2010) From tweets to polls: linking text sentiment to public opinion time series. In: ICWSM'11, Washington, DC, pp 122–129
31. Oghina A, Breuss M, Tsagkias M, de Rijke M (2012) Predicting IMDb movie ratings using social media. In: ECIR'12: proceedings of the 34rd European conference on advances in information retrieval
32. Rand WM (1971) Objective criteria for the evaluation of clustering methods. JASIS 66:846–850
33. Ratkiewicz J, Fortunato S, Flammini A, Menczer F, Vespignani A (2010) Characterizing and modeling the dynamics of online popularity. Phys Rev Lett 105(15):158701
34. Redmond U, Harrigan M, Cunningham P (2012) Identifying time-respecting subgraphs in temporal, networks. In: MUSE'12, pp 60–76
35. Rowe M, Angeletou S, Alani H (2011) Anticipating discussion activity on community forums. In 2011 IEEE 3rd international conference on privacy, security, risk and trust and 2011 IEEE 3rd international conference on social computing, IEEE, pp 315–322, Oct 2011
36. Vinh NX, Bailey J, Epps J (2009) Information theoretic measures for clusterings comparison : is a correction for chance necessary? In: Proceedings of the 26th annual international conference on machine learning–ICML '09. ACM Press, New York, pp 1–8, June 2009
37. Wang YC, Joshi M, Cohen W, Rosé C (2008) Recovering implicit thread structure in newsgroup style conversations. In: ICWSM '08, Seattle, WA. USA
38. Wolfe AP, Jensen D (2004) Playing multiple roles: discovering overlapping roles in social networks. In: ICML-04 workshop on statistical relational learning and its connections to other fields

39. Xi W, Lind J, Brill E (2004) Learning effective ranking functions for newsgroup search. In: SIGIR '04, Sheffield, UK, pp 394–401
40. Yang J, Leskovec J (2011) Patterns of temporal variation in online media. WSDM'11. ACM, Hong Kong, China, pp 177–186
41. Yu Z, Zhou X, Zhou L, Du K (2010) A hybrid similarity measure of contents for TV personalization. Multimedia Syst 16:231–241

Predicting Users Behaviours in Distributed Social Networks Using Community Analysis

Blaise Ngonmang, Emmanuel Viennet and Maurice Tchuente

Abstract Prediction of user behaviour in Social Networks is important for a lot of applications, ranging from marketing to social community management. This chapter is devoted to the analysis of the propensity of a user to stop using a social platform in a near future. This problem is called churn prediction and has been extensively studied in telecommunication networks. We first present a novel algorithm to accurately detect overlapping local communities in social graphs. This algorithm outperforms the state of the art methods and is able to deal with pathological cases which can occur in real networks. It is then shown how, using graph attributes extracted from the user's local community, it is possible to design efficient methods to predict churn. Because the data of real large social networks is generally distributed across many servers, we show how to compute the different local social circles, using distributed data and in parallel on Hadoop HBase. Experimentations are presented on one of the largest French social blog platforms, Skyrock, where millions of teenagers interact daily.

B. Ngonmang (✉) · E. Viennet
L2TI, Institut Galilée, Université Paris Nord-Sorbonne Paris Cité, F-93430, Villetaneuse, France
e-mail: blaise.ngonmang@univ-paris13.fr

E. Viennet
e-mail: emmanuel.viennet@univ-paris13.fr

B. Ngonmang · M. Tchuente
Université de Yaoundé I, UMI 209 UMMISCO, B.P. 337, Yaoundé, Cameroun

Faculté des Sciences, Département d'Informatique, LIRIMA, Equipe IDASCO, B.P. 812, Yaoundé, Cameroun
e-mail: maurice.tchuente@ens-lyon.fr

F. Can et al. (eds.), *State of the Art Applications of Social Network Analysis*, Lecture Notes in Social Networks, DOI: 10.1007/978-3-319-05912-9_6,

1 Introduction

Churn prediction is a well studied data mining task in the context of telecommunication networks [1]. It consists of predicting whether, in a near future, a user will leave his present operator for another one. With the multiplication of social network platforms that generally provide similar functionalities, a subscriber of a particular platform can decide to churn and switch to another network. This phenomenon was observed on several platforms when Facebook arrived on the French market. Churn is just an example of a user behaviour that an operator would like to predict in order to take some appropriate action (marketing, ads targeting, recommendation, content pushing...).

The behaviours of users are often social or viral in the sense that one user will influence or be influenced by his or her friends. It thus makes sense to study the local environment of a user in order to predict churn more accurately.

Community detection in social networks has attracted a lot of attention in the past years. The resulting communities can be used to understand the underlying network structure. We show in the present work how the community of a user can be used to predict if he will become inactive (i.e. he will churn) in a near future.

Hereafter, we focus on local rather than global community detection. Indeed, methods for global community detection like those described in [2] generally assume that one can access the whole network. This assumption is not realistic for very large and dynamic networks, because it would induce too large computing times. Furthermore, existing global community detection methods like Louvain's algorithm [3] (one of the best methods to efficiently partition graphs with millions of nodes) usually produce very large communities that are not useful in practice. This is a consequence of the optimization of global criteria like modularity [4] . To overcome this drawback, several local community detection methods have been introduced: starting from a node, one explores at each step the neighbourhood of the current local community, and include the external node that produces the maximum gain for the objective function used by the method. Examples of such methods are presented in [5–8]. In this chapter we will focus on these local methods.

The contribution of *local community-based* attributes for churn prediction will be illustrated here on a dataset provided by Skyrock (http://www.skyrock.com). Skyrock is a large online blog network where users can establish friendship relations. It is then possible to compute the communities in this friendship network. The aim of this work is to show that the local communities of users in real friendship social networks such as Skyrock, produce relevant attributes for the construction of a supervised learning model for churn prediction.

Note that, in this study, we do not use the contents posted by the users, and don't analyse their behaviours (except for absence or presence of connections to the platform). We only focus on social ties. Better results could in principle be obtained by taking into account these indicators, but the analysis becomes more complex.

Implementing Social Networks Analysis methods on a real network application is not trivial; the data is huge, the processing environment is complex and results

are needed in real time. Moreover, the data of real large social networks is usually distributed across many physical servers. We then need methods that can deal with distributed data and that can take advantage of all these servers for computation. In this chapter we propose an approach to deal with distributed data using Hadoop HBase framework.

The rest of this article is organized as follows: in Sect. 2, we present the churn prediction problem in the context of social networks; Sect. 3 recalls an algorithm we have proposed earlier for local community identification in social networks; Sect. 4 presents our framework for distributed Social Network Analysis (SNA); Sect. 5 presents the dataset and the methodology for the construction of the network; Sect. 6 presents the experiments performed and the relevant results; and finally Sect. 7 presents some conclusions and discussions.

2 Churn Prediction in Social Networks

The term churn is derived from *change* and *turn*. It means the discontinuation of a contract. In business applications, it corresponds to customer loss. In telecommunications, a subscriber is said to have churned when he leaves one carrier to move to another [9], and this phenomenon has been extensively studied [10, 11]. Indeed, it is generally admitted that retaining an existing customer is less expensive than winning a new one [12]. As a consequence, telecommunication companies tend to move important marketing efforts from customers acquisition to customers retention. Churn prediction is therefore an important task for Customer Relationship Management (CRM).

To predict churn, dozen to hundred of attributes are generally derived from the customer's profiles and service usages. These features are then used to build a statistical model for churn prediction based on supervised learning [9]. This pure *feature-based* churn prediction has the limitation that it does not take into account the social relations between subscribers.

Indeed, in social networks, social ties or links are also relevant to churn prediction [11] because people form communities and are more active with members within their local communities than with members outside their local communities. It follows that if many members of a user community stop using a service, the number of persons with whom this user can interact through that service decreases and the probability that he also churns gets higher. The local community of a user can therefore be mined to provide *community-based* attributes for churn explanation.

The analysis carried out in [11] is one of the first in that direction and explores the propensity of a subscriber of a mobile operator to churn, depending on the number of friends that have already churned. The method proposed by Dasgusta et al. is based on a diffusion process and can be summarized as follows:

- the initial seeds set contains the known churners of the past period;
- at each iteration the seeds try to activate their neighbors based on a diffusion function;

- all the new activated nodes are included in the new seeds set;
- the process stop when there is no new activated node and there is not a significant change in the level of activation for each node.

The limitation of this approach is that the notion of community is implicitly defined by the diffusion process. Moreover, it cannot easily consider other available information (socio-demographic information, age, gender, adresse, profession,...) when they are available.

Other studies have also been conducted in P2P networks. For example, authors of [13] have studied the bias induced by the length of the observation period while predicting the churn in P2P networks.

3 Local Community Identification in Social Networks

In this section, we recall some basic results published in [8] on the identification of local communities in social networks. A network is represented by an undirected and unweighted graph $G = \langle V, E \rangle$, where V is the set of nodes and E is the set of edges. A neighbour of node u is a vertex v such that $(u, v) \in E$. $\Gamma(u)$ denotes the set of neighbours of u. Let D be a subset of V. An edge $e = (u, v)$ is internal to D if both ends u and v are in D. An outgoing link is an edge $e = (u, v)$ that has only one end in D.

The density δ of links present in a graph is a generic notion that refers to the number of links divided by the number of nodes to which they are incident. The internal density δ_{in} corresponds to the number of internal links of a sub-graph divided by the number of vertices. Similarly, the external density δ_{out} corresponds to the number of outgoing links divided by the number of nodes.

A community of a network G is a subset of nodes D such that δ_{in} is high, and δ_{out} is low. In the case where the network is not fully known, the community produced by exploring G starting from a node n_0 is called the local community of n_0.

3.1 General Greedy Scheme for Community Detection

Most existing algorithms for local community identification use a greedy scheme: initially, the local community D contains only the starting node n_0 and the quality of this initial community is 0. At each step, the external node that maximizes the quality function Q used by the algorithm is considered. If its inclusion into D increases the quality criterion Q, then it is added to D, and the quality Q of the community is updated. This procedure is repeated until there is no more external vertex whose inclusion into D increases the quality Q. At the end of the algorithm, D contains the local community of n_0.

3.2 Quality Functions for Local Community Identification

In the rest of this article, D denotes a local community, B is the set of nodes of D that have at least one neighbour out of D and S is the set of nodes out of D that have at least one neighbour in D.

Clauset [14] introduces the idea that nodes on the border of a community (nodes of B) must have more links with D than with S. The local modularity of D is then defined by formula (1):

$$R = \frac{B_{in}}{B_{in} + B_{out}} \tag{1}$$

where $B_{in} = \sum_{u \in B} |\Gamma(u) \cap D|$ is the number of links between B and D, and $B_{out} = \sum_{u \in B} |\Gamma(u) \cap S|$ is the number of links between B and S.

Luo et al. [15] has proposed another quality function that takes into account all the internal links rather than just those edges that link B to D. This idea leads to the quality function defined by:

$$M = \frac{D_{in}}{D_{out}} \tag{2}$$

where $D_{in} = \sum_{u \in D} |\Gamma(u) \cap D|$, and $D_{out} = B_{out}$.

Chen et al. [6] have proposed a new quality function L and a new method. This method introduced a first innovation: it considers the densities of intra-community edges and outer edges and not their numbers. More precisely the density of intra-community links L_{in} is defined according to the expression:

$$L_{in} = \frac{\sum_{i \in D} |\Gamma(i) \cap D|}{|D|} \tag{3}$$

Similarly, the density of external links is defined by:

$$L_{ex} = \frac{\sum_{i \in B} |\Gamma(i) \cap S|}{|B|} \tag{4}$$

Chen et al. then use the ratio of these two densities to define a quality function:

$$L = \frac{L_{in}}{L_{ex}} \tag{5}$$

To avoid the improper inclusion of vertices into the local community, an increase of the quality function L does not induce the automatic inclusion of a new node into D. More precisely, let L'_{in} and L'_{ex} denote the new densities if n_i is added to D. A node is included only if $L'_{in} > L_{in}$.

At the end of the main loop, i.e. when there is no extra outside node that produces a positive gain when added to D, the algorithm reviews the nodes of D. Each node

of D is removed and a test to add it is carried out. During this confirmation stage, a node is maintained in D only if $L'_{in} > L_{in}$ and $L'_{ex} < L_{ex}$.

We recently proposed [8] to improve the algorithm of Chen et al. by adding at each step, and at the same time, all the nodes that maximize the quality function, instead of selecting one at random. We also have proposed a new quality function T which favours the nodes that are closer to the starting node. Hereafter, the corresponding algorithm is named "BME" (after the author's firstnames). In this approach the contribution of internal links is given by:

$$T_{in} = \frac{\sum_{i \in D} \frac{|\Gamma(i) \cap D|}{(1+d_i)}}{D} \tag{6}$$

where d_i is the length of the path from the starting node n_0 to node i in the tree generated by the algorithm. In this expression, internal links that are close to the starting node are favored by the multiplicative factor $1 + d_i$. For external links, the contribution is defined by:

$$T_{ex} = \frac{\sum_{i \in D} |\Gamma(i) \cap S|(1 + d_i)}{D} \tag{7}$$

In this expression, external links that are far away from the starting node are penalized by the multiplicative factor $1 + d_i$. This leads to the quality function:

$$T = \frac{T_{in}}{T_{ex}} \tag{8}$$

We have shown in [7] that BME performs better than the Chen et al. [6] method which was from our best knowledge the best method at that time.

The time complexity of the algorithms derived from Chen et al.'s method, depends only on the average degree of the network, the size of the local community found and the number of neighboring nodes of that community. It is in the worst case $O(|D|d|S|)$, were $|D|$ and $|S|$ are usually very small compared to the size of the whole network.

3.3 Local Overlapping Communities

A natural idea for identifying overlapping communities is to take an algorithm A for local community identification and apply the scheme of Algorithm 1. In this scheme, A corresponds to one pass through the loop, V is the set of nodes confirmed in the second phase of the algorithm A and LocalCom is the table of the local overlapping communities found, indexed by idcom. After each execution of A, the links that are internal to $V \setminus \{n_0\}$ are deleted.

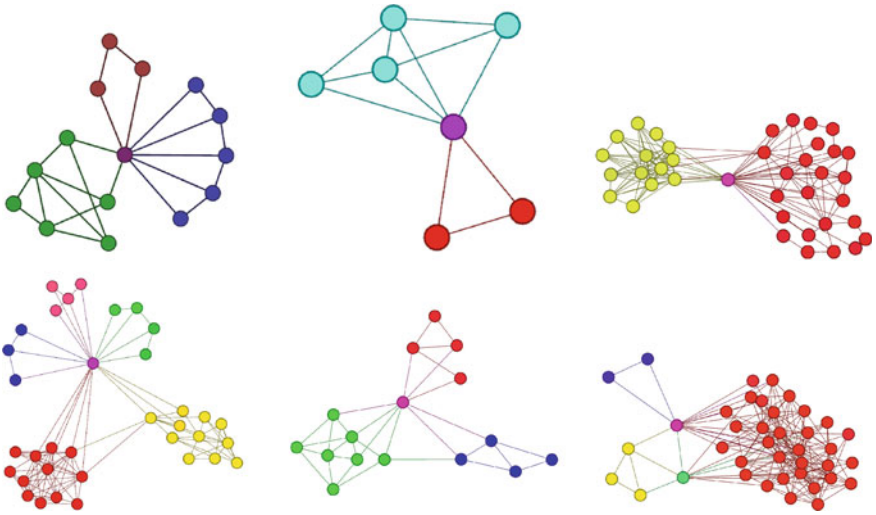


Fig. 1 Some examples of local communities found by algorithm IoLOCo in Skyrock dataset. The starting node is at the center. We drew only the internal nodes and links. We see that in most cases, the starting node belongs to many local communities and that some time some others nodes also belong to more than one communities (see the *bottom rightmost* figure.)

Even in the case of non-overlapping communities, the application of IOLoCo achieves the best results [8]. Figure 1 presents some communities found IOLoCo with In the next sections, IOLoCo will be used to compute communities for churn prediction.

Algorithm 1 Identification of Overlapping Local Communities

Algorithm: IOLoCo
Input: a graph G and a starting node n_0.
Output: a table $LocalCom$[..] of local overlapping communities of n_0
idcom = 0
Initialize $LocalCom$ with the empty table
Repeat
V = the local community of n_0 produced by algorithm A
if $n_0 \in V$ **or** $|\Gamma(n_0) \cap V| >= \sum_{i \in V} \frac{|\Gamma(i) \cap V|}{|V|}$ **then**
$LocalCom$[idCom] $= V$
idcom = idcom + 1
end if
Mark all the internal links of $V \setminus \{n_0\}$ as "deleted"
Until ($\Gamma(n_0) = \emptyset$)
Return $LocalCom$

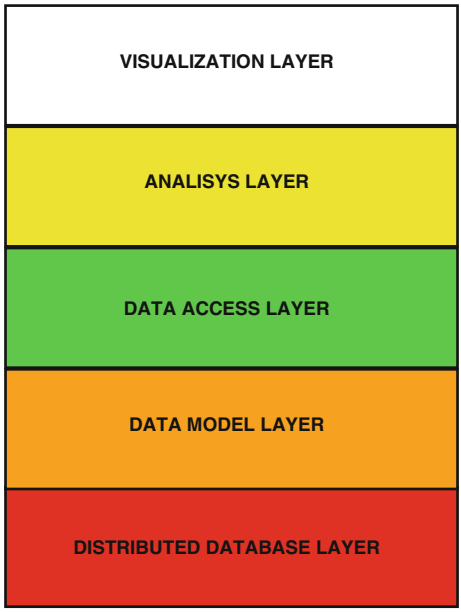


Fig. 2 Distributed SNA model

4 A Model for Social Network Analysis with Distributed Data

The data of real large social networks is usually distributed across many physical servers. It is then necessary to develop methods that can access these data efficiently. In this section, we describe a general framework to perform social network analysis (SNA) in general and the computation of local circles (local communities and neighbourhood) in particular, in the case where the data are distributed across many servers. This situation is always the case for very large and dynamic on-line social networks.

The proposed model is composed of the following layers : the Distributed Database Layer (DDL), the Data Model Layer (DML), the Data Access Layer (DAL), the Analysis Layer (AL) and the Visualization Layer (VL). Figure 2 presents all these layers.

4.1 Distributed Database Layer

First of all, the persistence and the distribution of the data need to be described. For this purpose we define a Distributed Database Layer. For this layer any distributed database can be used so that the system can be able to scale horizontally.

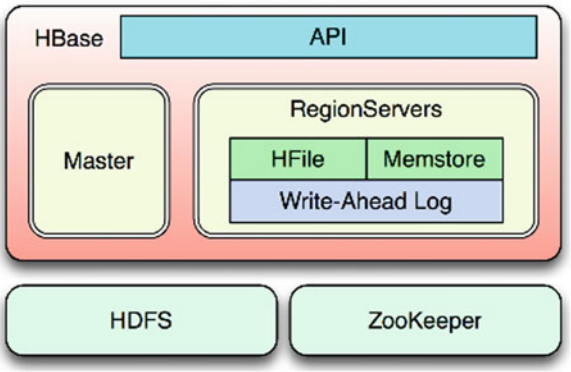


Fig. 3 HBase architecture

The implementation presented here uses HBase, the Hadoop Database [16] that is an open-source, distributed, versioned and column-oriented [17]. Figure 3 presents an overview of HBase architecture:

- the data is stored using the Hadoop Distributed File System (HDFS) [18].
- Master is responsible for assigning regions to RegionServers and uses Apache ZooKeeper, a reliable, highly available, persistent and distributed coordination service.
- the RegionServers manage the regions and the storage of data on HDFS.
- API is the component that enables to query the database using programming languages.

More information about HBase can be obtained in [16] or by browsing the official website: http://hbase.apache.org/.

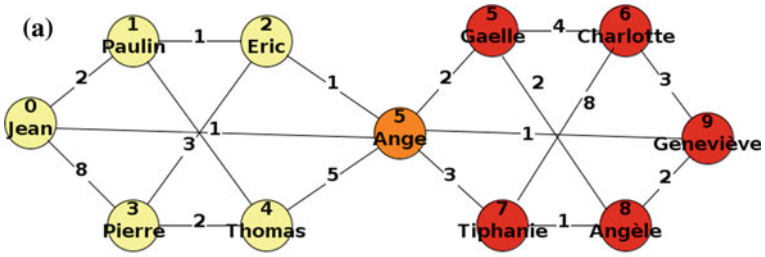
4.2 Data Model Layer (DML)

After the choice of the distributed database, a data model for the representation of a social network must be defined. The Data Model Layer (DML) is a table where each row corresponds to a node attribute or to an edge:

- an edge (u,v) with label l is represented by (u, Neigh, v, l)
- an attribute of node u with type T and value V is represented by (u, Attrib, T, V).

Figure 4 presents an example of social network and the four rows that represent the information concerning node 0.

The basic operations implemented are the following: add or remove a node, add or remove an edge, get the neighbours of a node etc.



(b)

Row ID	Column Family	Column qualifier	Value
0	Neigh	1	2
0	Neigh	3	8
0	Neigh	5	1
0	Attrib	Name	Jean

Fig. 4 A toy social networks (**a**) and the representation of node 0 in the proposed model (**b**)

4.3 Data Access Layer

Having defined a distributed database and a model to represent a social network in this database, it is now time to show how to retrieve and update data of a social network. For this purpose three classes and an interface are defined. The classes Graph, Node and Edge are self explanatory. The interface DataAccessInterface provides the access point to query the distributed database using the previous sub-mentioned classes as inputs and/or results. Figure 5 presents the classes and the relation between them whereas Fig. 6 presents the interface and a concrete class that implements it following a UML representation.

With this model all the operations of social network analysis can be defined efficiently. The operations provided by the DataAccessInterface are:

- boolean insertNode(Node node): insert a node to the database
- Node getNode(): get a node from the database
- boolean insertNode(Node node): insert a node to the database
- Edge getEdge(): get an edge from the database
- long vcount(): return the number of nodes
- long ecount(): return the number of edges
- boolean beginBatch(): begin a transaction mode
- boolean endBatch(): end a transaction mode
- boolean close(): close the connection to the database.

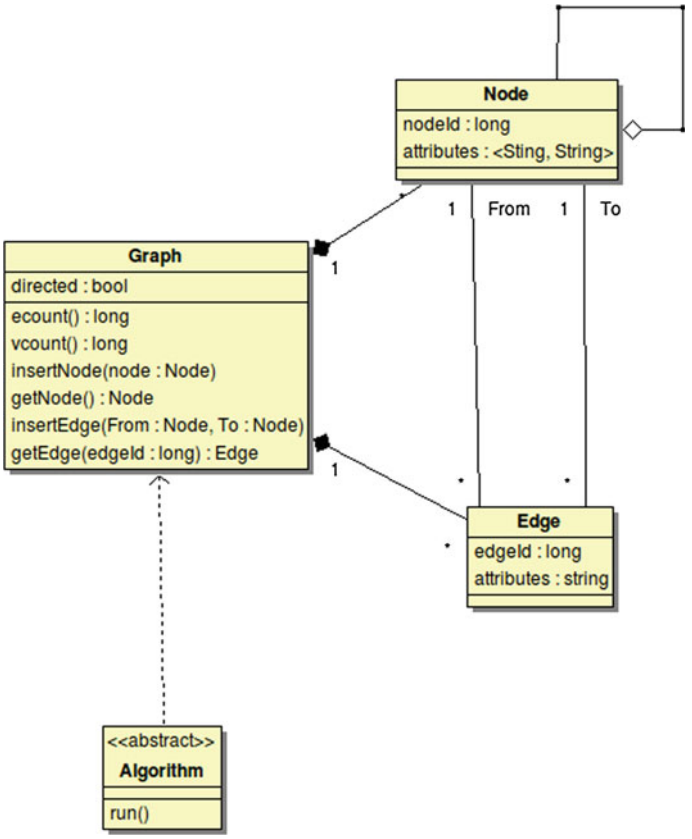


Fig. 5 SNA layer implementation example

4.4 Upper Layers

Given the description of the bottom layers, the upper ones (AL and VL) are able to query the Data Access Layer, interact with the data and process them.

The implementation of these layers consists of different algorithms that solve problems for social network analysis or visualisation.

Examples of tasks that can be performed are the computation of local communities and other social circles like neighbourhoods. In fact the distributed algorithms are the same as in the case where the data is located on the computation node. We only need to replace each sequential instruction that gives access to a node or to an edge, by the corresponding one that is distributed. Since the computation of each social circle is independent of the others, the total computation time of the distributed program decreases linearly with the number of processing nodes. We have conducted experiments with one to ten processors and the results are reported in Fig. 7.

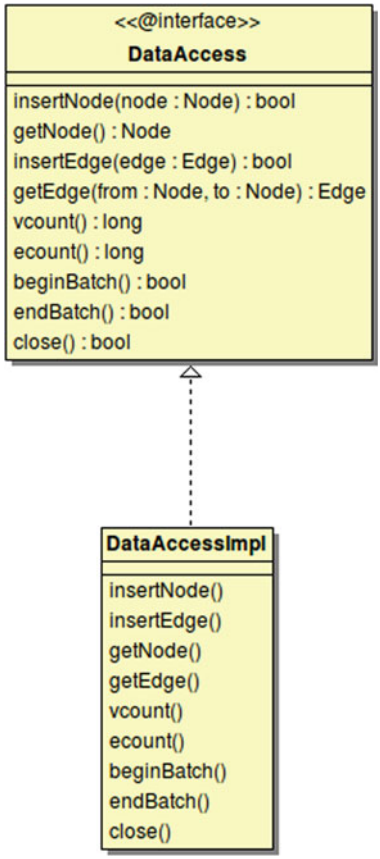


Fig. 6 Data access layer implementation example

5 Dataset Description and Attributes Extraction

Skyrock (http://www.skyrock.fr) is a social platform where users (aka "skynautes") can (among other actions) create blogs, add comments, create tags and define explicitly friendship relations. The global dataset of Skyrock has 31.3×10^6 nodes (the user's profiles) and 1.17×10^9 links (the friendship relations). Figure 8 shows a screenshot of the Skyrock Social Network.

The dataset used for the experimentations is constructed as follows (Fig. 9): from the global network, the subgraph of active users in March 2011 is extracted, because the churn prediction concerns only active users that become inactive. A user is active if he has made at least one connexion to the platform during the considered period of time. In the following, only this graph formed by active users is considered

After that, all the nodes that have more than 5,000 friends are removed because they generally represent celebrities ("mega-hubs" in the network) with abnormal behaviour. This new subgraph has 2.24×10^6 nodes and 127.428×10^6 links compared

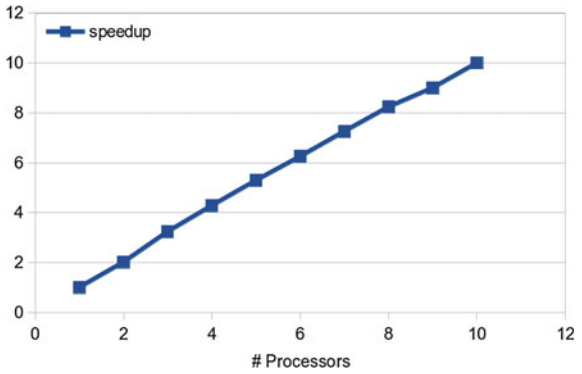


Fig. 7 Speedup results: the computation time linearly decreases with the number of processors

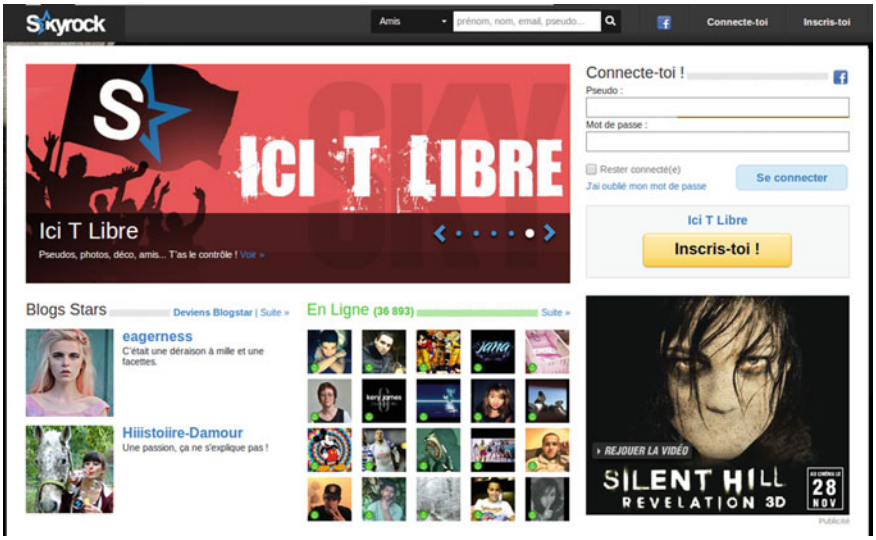


Fig. 8 The skyrock social network. The main page where you can login or subscribe to start a social experience with your friends and other users

to the 31.3×10^6 nodes and 1.17×10^9 friendship links of the whole network. Figure 10 presents the degree distribution of the nodes of this network. Not surprisingly, the degrees seem to follow a power law.

Local communities are then computed using the algorithm IOLoCo presented in Sect. 3. The distribution of sizes of these local communities is shown in Fig. 10.

For completeness the global communities detected by the Louvain algorithm [3] were also computed. It is a global method that can compute communities efficiently in very large networks and is based on an heuristic optimization to maximize the modularity of the partition.

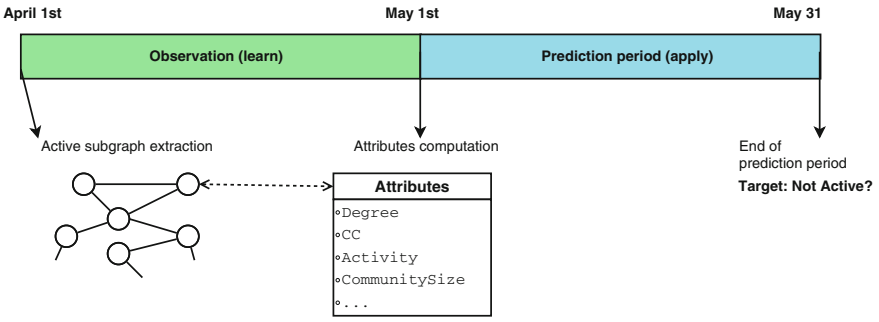


Fig. 9 Experimental setup used for churn prediction

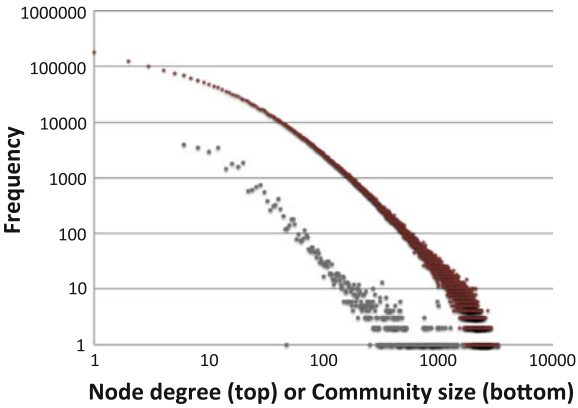


Fig. 10 Distributions of nodes' degrees and of sizes of local communities found in Skyrock network (log-log scale). The *top curve* corresponds to *nodes*' degrees and exhibits a typical scale-free distribution. The *bottom curve* corresponds to the size of local communities: most are small, but some *nodes* form larger communities, and the log-log curve is *quasi-linear*

Then, some simple attributes are defined to characterize a node and its vicinity. We consider only very simple attributes (degree, community size, proportion of active nodes, etc.) which can be computed quickly and have straightforward interpretation. Some of these attributes are related to the starting node and some of its social circles such as its local community, the first neighborhood , the second neighborhood (see Fig. 11) and the Louvain's community. All these attributes are presented in Table 1.

Then 50,000 users are selected at random to create the learning dataset. This dataset has been divided into training (66 %) and test (34 %) sets. A supervised learning algorithm is trained and estimated on this dataset.

The results of attribute extraction using the methods introduced above, can now be presented. It's worth looking at the sizes of the sets of nodes used to compute the attributes: the *local community* size (attribute *ComSize*) ranges from 1 to 547 with a mean of 21. The first neighborhood size (*NeigSize*) is large, ranging from 1 to 4,644 with a mean of 669. The second neighbourhood size (*Neig2Size*) is even larger (4–42,3089 with a mean of 79,702). Finally, the sizes of global communities (*LouvainSize*) are as usual very heterogeneous, ranging from 2 to 511,457 with a mean of 359,625. Thus, the local communities are by far the smallest sets of nodes

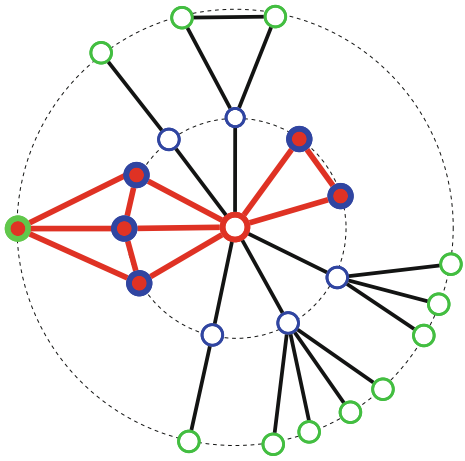


Fig. 11 A central node and its first neighbourhood (*inner circle*), its second neighbourhood (*outer circle*) and its local community (*nodes in bold*)

that we will consider. This leads to a faster algorithm and allows us to focus on the most important neighbours.

In this application, the churn problem consists, starting from a set of active users by the end of March, to observe their activity during the month of April in order to predict which users will churn in May. The objective is to show the contribution of the (local) community of a node in churn prediction. This is presented in the next section.

6 Experimentations and Results

Various models, based on different subsets of attributes, were built. A tool provided by KXEN (www.kxen.com) (that is known to be very efficient [19, 20]) is used to identify the most relevant attributes, amongst the attributes enumerated above. This is done using their contributions to the underlying model. In this context, the output of KXEN is a ranking of the attributes considered as explanotary variables.

6.1 Experiments with Various Attributes Sets

All attributes are numerical and have been normalized between −1 and 1. The experiments are based on standard Support Vector Machines—a well established statistical classifier. The ***LIBSVM*** software [21] is used.

The Radius Basis Function (RBF) kernel has been chosen. This kernel non-linearly maps samples into a higher dimensional space and allows to build non-linear decision frontiers. Note that the linear kernel is a special case of RBF [22].

The parameter C for each model was fixed to 1 and the parameter γ to the inverse of the number of features. In order to compensate the difference between the number

Table 1 Some attributes for churn prediction

	Attribute name	Description
1	Degree	The degree of the node
2	CC	The local clustering coefficient of the node
3	Activity	The number of time the node has made a connexion during the learning month
3	DaysAfterLastCon	The number of days since the last connexion of the node during the learning month
4	LocalComSize	The size of the local community i.e. the number of nodes of the local community
5	LocalInProp	The internal proportion i.e. the proportion of local community's node directly connected to the starting node
6	LocalAvgDegree	The average degree of the nodes inside the local community
7	LocalPropInact	The proportion of nodes inside the local community that are already inactive
8	LocalAvgAct	The average activity for the nodes of the local community
9	NeigSize	The size of the first neighborhood
10	NeigAvgDegree	The average degree of the first neighborhood
11	NeigPropInact.	The proportion of nodes inside the first neighborhood that are already inactive
12	NeigAvgAct	The average activity for the nodes of the first neighborhood
13	Neig2Size	The size of the second neighborhood
14	Neig2AvgDegree	The average degree of the second neighborhood
15	Neig2PropInact	The proportion of nodes inside the first neighborhood that are already inactive
16	Neig2AvgAct	The average activity for the second neighborhood
17	LouvainSize	The size of the Louvain's global community the node belongs to
18	LouvainAvgDegree	The average degree of the Louvain's global community the node belongs to
19	LouvainPropInact.	The proportion of nodes inside the Louvain's global community the node belongs to that are already inactive
20	LouvainAvgAct	The average activity for the Louvain's global community the node belongs to
21	Not active?	The target attribute we want to predict

of examples in each class, we weighted the positive class according to the proportion of churners.

To train the different models, the following sets of attributes were used:

Table 2 Evaluation with support vector classifiers

Method	Avg #nodes used	AUC
All	431978	0.855
All without Louvain's global community	72353	0.854
Node & local community	21	0.832
SPA method [11]	–	0.829
Node & second neighborhood	71734	0.826
Node & first neighborhood	598	0.824
Node & Louvain's global community	359625	0.823
Node only	1	0.815
Local community only	20	0.727
Second neighborhood only	71733	0.699
First neighborhood only	598	0.649
Louvain community only	359624	0.635

1. all the attributes;
2. the node attributes;
3. the node attributes and the first neighborhood attributes;
4. the node attributes and the second neighborhood attributes;
5. the node attributes and the local community attributes;
6. the node attributes and the Louvain community attributes;
7. all the attributes except Louvain community attributes;
8. the first neighborhood attributes;
9. the node second neighborhood attributes;
10. the local community attributes;
11. the Louvain community attributes.

The models were trained and then applied to the test set in order to compute the area under ROC Curve (AUC) was computed. For sake of comparison, the AUC obtained with the method by Dasgusta et al. (SPA) [11] is also displayed.

The Receiver Operating Characteristic (ROC) curve is a plot of the *true positive rate* against the *false positive rate* for the different possible *decision thresholds*. The area under this curve represents how well the model can separate the two classes.

Table 2 presents the results of the experiments using the above performance indicators. The results are ranked by the value of AUC which is the most relevant performance indicator because the classes are unbalanced.

One can observe from Table 2 that the best model is obtained by using all the attributes. This illustrates the well known robustness of SVM models, which are able to cope with a lot of variables. The second model is the one with all variables except those extracted from global communities. This is not a surprise, because global communities are too large for this application and thus are not able to capture the local dynamics of the network. The local circle that gives the best prediction is the local community.

The second most performant model (bolded in Table 2) is based only on the attributes computed from the user and its local community. On average, this model considers the state of 21 nodes, while the average degree of the nodes is 669: the local community allows to focus on the most important neighbours. In an application where local communities are incrementally computed and maintained, this leads to very efficient estimators.

The relevance of the local community is confirmed by the observation that when the nodes attributes are not considered, the best performance is achieved using the local community only. This shows clearly that the most important contribution to churn prediction is provided by the local community.

One can observe that the method by Dasgupta et al. (SPA) [11] has a performance that is close to what is obtained with the local community. However, contrary to SPA, the method proposed here can easily take into account non graph-based attributes such as age, gender, profession and salary. in the statistical model. Moreover, the approach proposed in this chapter is more flexible and modular with a clear separation between the computation of the local community and the estimation of the outcome: both the community detection algorithm and the statistical model could be replaced by other versions adapted to specific applications.

It can be seen that the second neighbourhood is better than the first one, but leads to an algorithm that is less accurate and much slower than the one based on the local community.

6.2 Ranking of Attributes

The second experiment uses InfiniteInsight, a tool developed by KXEN (a data mining company which provides tools that can handle millions of records in an effective way). The K2C/K2R (Kxen Consistent Coder and Robust Regression) modules are able to select the most relevant attributes and compute their contribution to the model. With these modules, one can built a model using all the attributes except those related to the activity of the node itself. The aim of this test is to identify the topological attributes that have the most important contribution. The results are shown in Fig. 12. It can be seen from Fig. 12 that the most relevant topological attribute for churn prediction is *PropInact*: the proportion of inactive members of the local community. This result reinforces the intuition that nodes are highly influenced by the behaviours of local community members: the most important explicative factor of churn is the number of friends in the local community that churned during the period of observation. This generalizes the result of [11] where it was shown that the number of friends (neighbours with strong links) that have churned, has a great impact on churn prediction.

Figure 12 also shows that the second most relevant topological attribute is the proportion of nodes that are inactive in the second neighbourhood (*Neig2InactProp*). This is consistent with the previous test : after the local community, the second neighbourhood produces the most relevant attributes.

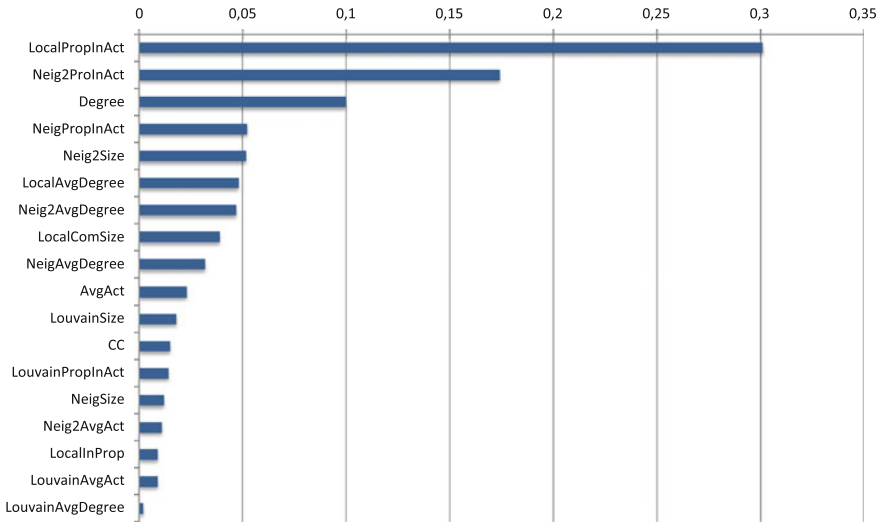


Fig. 12 Variable contributions to the K2C/K2R model

7 Conclusion and Discussions

In this chapter, we have studied the churn prediction problem in social networks. It is shown that local communities that are known to be quickly and accurately computable, can be used to extract attributes that are relevant for churn. An approach has been proposed to deal with real Social Networks that are usually distributed on several servers. A successful implementation using the Hadoop Hbase framework has been presented.

We are currently working on an extension that takes into account nodes attributes and the dynamics of local circles.. Indeed, users are continuously generating contents (likes, tags, posts, comments) that, although complex and noisy, can be used to improve the prediction of users' behaviours. On the other hand, the analysis of the dynamics of local communities can clearly help to better understand the users' interactions and future behaviours.

Acknowledgments This work has been partially supported by the ANR projects Ex DEUSS, DGCIS CEDRES, and by FUI project AMMICO.

References

1. Richter Y, Yom-Tov E, Slonim N (2010) Predicting customer churn in mobile networks through analysis of social groups. In: Proceedings of the 10th SIAM international conference on data mining, Apr 2010
2. Fortunato S (2010) Community detection in graphs. Phys Rep 486:75–174

3. Blondel VD, Guillaume JL, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. J Stat Mech: Theory Exp 10008
4. Good BH, De Montjoye YA, Clauset A (2010) Performance of modularity maximization in practical contexts. Phys Rev E 81(4):046106
5. Bagrow JP (2008) Evaluating local community methods in networks. J Stat Mech 05:05001
6. Chen J, Zaiane OR, Goebel R (2009) Local communities identification in social networks. In: ASONAM, pp 237–242
7. Ngonmang B, Tchuente M, Viennet E (2011) Identification de communautés locales dans les réseaux sociaux. In: AGS, conférence CAP, pp 16–27, Mai 2011
8. Ngonmang B, Tchuente M, Viennet E (2012) Local communities identification in social networks. Parallel Process Lett 22(1), Mar 2012
9. Hwang H, Jung T, Suh E (2004) An LTV model and customer segmentation based on customer value: a case study on the wireless telecommunication industry. Expert Syst Appl 26(2):181–188
10. Mozer M, Wolniewicz RH, Grimes DB, Johnson E, Kaushansky H (1999) Churn reduction in the wireless industry. In: NIPS, pp 935–941
11. Dasgupta K, Singh R, Viswanathan B, Chakraborty D, Mukherjea S, Nanavati A, Joshi A (2008) Social ties and their relevance to churn in mobile telecom networks. In: Proceedings of the 11th international conference on extending database technology, EDBT '08, pp 668–677
12. Hadden J, Tiwari A, Roy R, Ruta D (2007) Computer assisted churn management: stat-of-the-art and futur trends. Comput Oper Res 34(10):2902–29177
13. Good BH , De Montjoye YA, Clauset A (2010) Performance of modularity maximization in practical contexts. Physical Review E 81(4):046106
14. Clauset A (2005) Finding local community structure in networks. Phys Rev 72:026132
15. Luo F, Wang JZ, Promislow E (2006) Exploring local community structure in large networks. In: WI'06, pp 233–239
16. George L (2011) HBase: the definitive guide. O'REILLY, California
17. Chang F, Dean J, Ghemawat S, Burrows M, Chandra T, Fikes A (2008) Bigtable: a distributed storage system for structured data. ACM Trans Comput Syst 26(2):1–26
18. White T (2010) Hadoop: the definitive guide, 2nd edn. Yahoo Press, California, Oct 2010
19. Chapus B, Fogelman Soulié F, Marcadé E, Sauvage J (2011) Mining on social networks. Statistical learning and data science. In: Gettler Summa M, Bottou L, Goldfarb B, Murtagh F (eds) Computer science and data analysis series. CRC Press, Chapman & Hall, London
20. Fogelman Soulié F, Marcadé E (2008) Industrial mining of massive data sets. Mining massive data sets for security advances in data mining, search, social networks and text mining and their applications to security. In: Fogelman-Soulié F, Perrotta D, Pikorski J, Steinberger R (eds) NATO ASI series. IOS Press, Amsterdam
21. Chang CC, Lin CJ (2011) LIBSVM: a library for support vector machines. ACM Trans Intell Syst Technol 2:27:1–27:27. Software available at http://www.csie.ntu.edu.tw/cjlin/libsvm
22. Kerthi S, Lin C (2003) Asymptotic behaviors of support vector machines with Gaussian Kernel. Neural Comput 15:1667–1689

What Should We Protect? Defining Differential Privacy for Social Network Analysis

Christine Task and Chris Clifton

Abstract Privacy of social network data is a growing concern that threatens to limit access to this valuable data source. Analysis of the graph structure of social networks can provide valuable information for revenue generation and social science research, but unfortunately, ensuring this analysis does not violate individual privacy is difficult. Simply anonymizing graphs or even releasing only aggregate results of analysis may not provide sufficient protection. Differential privacy is an alternative privacy model, popular in data-mining over tabular data, that uses noise to obscure individuals' contributions to aggregate results and offers a very strong mathematical guarantee that individuals' presence in the data-set is hidden. Analyses that were previously vulnerable to identification of individuals and extraction of private data may be safely released under differential-privacy guarantees. We review two existing standards for adapting differential privacy to network data and analyze the feasibility of several common social-network analysis techniques under these standards. Additionally, we propose *out-link privacy* and *partition privacy*, novel standards for differential privacy over network data, and introduce powerful private algorithms for common network analysis techniques that were infeasible to privatize under previous differential privacy standards.

1 Introduction

Social networks are powerful abstractions of individuals and the relationships that connect them; social network analysis can be a very powerful tool. For example, understanding how well-connected a network is can aid in the development of

C. Task (✉) · C. Clifton
Department of Computer Science, Purdue University, West Lafayette, IN, USA
e-mail: ctask@purdue.edu

C. Clifton
e-mail: clifton@purdue.edu

F. Can et al. (eds.), *State of the Art Applications of Social Network Analysis*, Lecture Notes in Social Networks, DOI: 10.1007/978-3-319-05912-9_7,

word-of-mouth marketing campaign: How quickly will word of a product spread? Similar analysis is useful in epidemiology, predicting spread of a disease.

However, data about people and their relationships is potentially sensitive and must be treated with care to preserve privacy. Generally, social network graphs are anonymized before being made available for analysis. However, as several recent incidents have demonstrated, releasing even anonymized graphs may lead to re-identification of individuals within the network and disclosure of confidential information, with serious consequences for those involved. In 2007, Netflix released the Netflix Prize data-set, containing anonymized data about the viewing habits of its members, for public analysis by information retrieval researchers. Within a year, it had been demonstrated that wide-spread de-anonymization of individuals in the data-set was possible using public information from the Internet Movie Database [1]. By 2009, Netflix was involved in a lawsuit with one of its members who had been victimized by the resulting privacy invasion.

Privacy researchers have attempted to improve the security provided by graph anonymization techniques by adding noise to the node parameters and structure of the graph [2]. However, even a noisy graph structure with no node parameters whatsoever can be subject to deanonymization, particularly if an attacker has background knowledge of the network data [3]. For example, knowing the friendship relationships of a few individuals can make them identifiable in the released graph, leading to identification of their friends (and disclosure of information, such as other relationships, that those friends might not want publicly revealed.) As global social networks become more broadly accessible, these types of background knowledge are more readily available [3].

Differential privacy is a privacy standard developed for use on tabular data that provides strong guarantees of privacy without making assumptions about an attacker's background knowledge [4]. Differentially-private queries inject randomized noise into query results to hide the impact of adding or removing an arbitrary individual from the data-set. Thus, an attacker with an arbitrarily high level of background knowledge cannot, with a high degree of probability, glean any new knowledge about individuals from differentially-privatized results; in fact, the attacker cannot guess whether any given individual is present in the data at all.

While many of the privacy concerns associated with social-network analysis could be relieved by applying differential-privacy guarantees to common social-network analysis techniques, researchers have struggled to develop suitable adaptations of these techniques. Two principal difficulties arise: The adaptation of differential privacy from tabular data to network data, and the high sensitivity of social-network metrics to relatively small changes in the network structure.

Two models for applying differential privacy to social networks have arisen. *Node privacy* limits the ability of an attacker to learn any information about an individual, but at a high cost in added noise. *Edge privacy* protects against learning any particular relationship, but may allow learning general information about an individual. This chapter introduces *out-link privacy* and *partition privacy*, models for differential privacy that provide greater protection than edge privacy while allowing important

types of analysis that are not feasible under node privacy. The key contributions and outline of this chapter are as follows:

- A straightforward introduction to traditional differential privacy;
- A discussion of two known differential-privacy standards for network data, as well as the contribution of two new standards, *out-link privacy* and *partition privacy*, that provide strong privacy guarantees with the introduction of very small noise;
- A study of the feasibility of common social-network analysis techniques under differential-privacy;
- The contribution of two new algorithms satisfying *out-link privacy* that use ego-network style analysis to provide approximate results for queries that are too sensitive to perform under previous standards.
- A demonstration of the application *partition privacy* to a variety of contexts; *partition privacy* is a new approach that provides unprecedented levels of privacy with minimal noise, for studies that compare variables across multiple social networks. It allows the wide variety of techniques developed for traditional differential privacy to be applied to social-network privacy.

2 Traditional Differential Privacy

Differential privacy was developed by Cynthia Dwork and collaborators at Microsoft Research Labs [4]. It does not define a specific technique or algorithm; instead it states a mathematical guarantee of privacy that sufficiently well-privatized queries can satisfy. Consider a common sequence of events in social science research: a survey is distributed to individuals within a population; a subset of the population chooses to participate in the survey; individual information from the surveys is compiled into a data-set and some analysis is computed over it; the analysis may be privatized by the injection of random noise; and the final privatized result is released to the general public. Differentially-private queries offer a rigorous mathematical guarantee to survey participants that the released results will not reveal their participation in the survey.

We first introduce a few useful notations: I is set of individuals who contribute information to the data-set D_I (e.g., survey participants). The set of *all possible* data-sets is $\mathcal{D}$. We use $F : \mathcal{D} \rightarrow \Re^k$: to refer to the desired non-privatized analysis performed on a data-set and $Q : \mathcal{D} \rightarrow \Re^k$ to refer to the privatized implementation of F. We refer to the publicly released, privatized analysis results as R.

If R are the privatized query results that are released to the public, then R is the only evidence an attacker has about the nature of D_I. We introduce a possible-worlds model to understand how differential privacy works (see Fig. 1). We define D_I to be *true world* from which the analysis was taken. We also define any data-set that differs by the presence or absence of one individual to be a "neighboring" possible world: thus D_{I-Bob} is the neighboring possible world of D_I in which Bob chose to not participate in the survey.

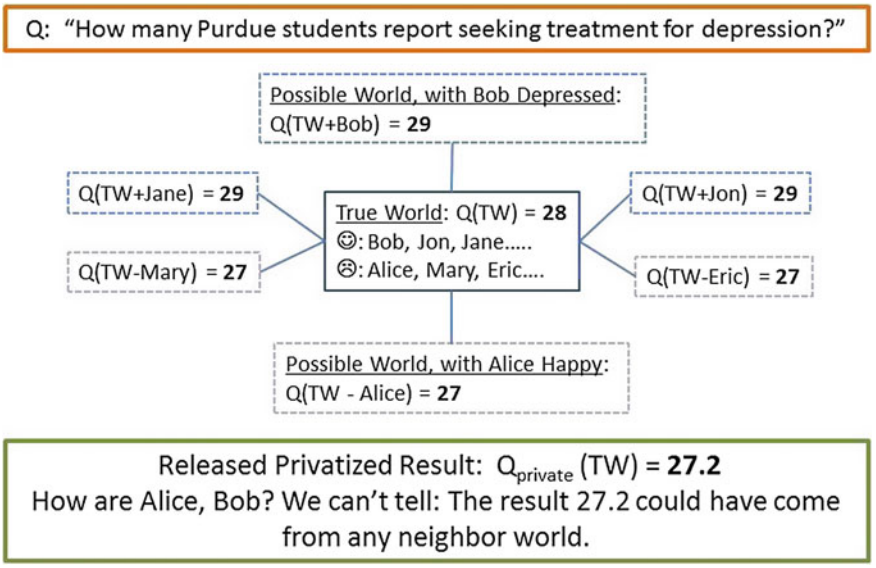


Fig. 1 Differential privacy adds noise to obfuscate individuals' affect on query results

We require that an attacker possessing the privatized results R be unable to determine whether or not Bob (or any other specific individual) took the survey, i.e., whether or not R are the results from an analysis of D_I or D_{I-Bob} (or, indeed, any neighboring world of D_I). Therefore, R should be a plausible result from any neighboring world of D_I.

Formally, D_I neighbors D_J iff $D_I = D_{J \pm x}$ for any x in the population, and:

Definition 1 A randomized query

$$Q : \mathcal{D} \rightarrow \Re^k$$

satisfies ϵ-differential privacy [4] if, for any two possible neighboring data-sets D_1, D_2 and any possible query result R:

$$\frac{Pr[Q(D_1) = R]}{Pr[Q(D_2) = R]} \leq e^{\epsilon}$$

Here ϵ is a small, positive value that controls the trade-off between privacy and accuracy, and is chosen by the person administering the privacy policy. To make the definition more intuitive, consider that if we set $\epsilon = ln(2)$, the above states that the result R is at most twice as likely to be produced by the true world as by any of its neighbors. Setting a smaller ϵ will provide greater privacy at the cost of additional noise, as we will demonstrate below.

The difference between the results from the true world D_1 and its neighbor D_2 is the difference the privatization noise will need to obfuscate in order for the privatized

results to not give evidence about whether D_1 or D_2 is the true world. The upper bound of this difference over $D_I \in \mathcal{D}$ is the *sensitivity* of query F.

Definition 1 The global sensitivity of a function $F : \mathcal{D} \rightarrow R^k = A$ is[1]:

$$\Delta F = \max_{D_1, D_2} \|F(D_1) - F(D_2)\|_1$$

over all pairs of neighboring data-sets D_1, D_2.

Intuitively, the sensitivity of a query is the *greatest* possible impact that adding or removing an arbitrary individual from the data-set can have on the query results, over *any* possible data-set. Suppose our analysis F asks two questions: "How many people in I are depressed?" and "How many people in I have fewer than 3 friends?" Then both answers can change by at most 1 when a single individual is added to or removed from I, and $\Delta F = 2$. If our analysis instead asks: "How many people in I are depressed?" and "How many people in I are happy?" then at most *one* answer can change by at most 1, and $\Delta F = 1$. Note that histograms, which partition the individuals of the data set into 'bucket' counts, have a sensitivity of 1: removing or adding an individual will change at most one bucket count by at most 1. This very low sensitivity makes histograms a useful tool in differentially private data-mining [4–6].

We can create a differentially private query Q by adding noise to F that is calibrated to cover up ΔF [4]:

Theorem 1 *If $F : \mathcal{D} \rightarrow \Re^k$ is a k-ary function with sensitivity ΔF then the function $F(D) + Lap^k(\Delta F/\epsilon)$ is ϵ-differentially private, where $Lap^k(\lambda)$ is a k-tuple of values sampled from a Laplacian random variable with standard deviation $\sqrt{2}\lambda$.*

The standard deviation of the Laplacian noise values is $\sqrt{2}\Delta F/\epsilon$. Thus the noise will be large if the function is very sensitive, or if ϵ is small. If we set $\epsilon = ln(2)$ on a query with sensitivity $\Delta F = 2$, the standard deviation of our added noise will be close to 4.

It is important to note that ΔF is an upper bound taken across *all possible* pairs of neighboring data-sets; it is independent of the true world. Intuitively, this is necessary because noise values which are dependent on the nature of the true world may introduce a privacy leak themselves. For example, when querying the diameter of a social network, if Alice forms the only bridge between otherwise unconnected subgraphs in the true world, removing her node and edges from the data-set causes a difference of ∞ in the graph diameter. Noise values calibrated to this true world must be arbitrarily large (and, in fact, will obliterate the utility of the result). However, consider a neighboring *possible* world including Bob, who forms a second bridge between the subgraphs (see Fig. 12); if this possible world were the true world, the difference in diameter caused by adding or removing a node would be finite, and if we calibrated the noise to that difference, it would be relatively small. If we chose

[1] The L_1-norm of $x \in \Re^n$ is defined as $\|x\|_1 = \Sigma_{i=1}^n |x_i|$.

our noise values based on the true world, an attacker could easily determine whether or not Bob was in the network: a result of $R = 300{,}453.23$ would imply Bob was absent, while the result $R = 4.23$ would indicate that Bob was present. To prevent this, global sensitivity is based on the worst-case scenario for the query, across all *possible* data-sets. In this case, this implies that diameter is a query too sensitive to be feasibly privatized using traditional differential privacy.

2.1 Smooth Sensitivity

Several sophisticated privatization techniques exist which do calibrate noise to the true data-set rather than using the worst-case upper-bound offered by global sensitivity. Consider an actual data-set D_{June12}; the *local sensitivity* of a function F on the data D_{June12} is the maximum change in F caused by removing or adding an individual from D_{June12}, analogous to computing the global sensitivity with D_1, D_2 restricted to D_{June12} and its neighboring possible worlds. In the example above, $diameter(G_{bob})$'s local sensitivity is small, while the local sensitivity of its neighbor $diameter(G_{alice})$ is very high: this jump in local sensitivities is what causes the threat to privacy described above. Since G_{alice} is created by removing one individual from G_{bob}, we will refer to G_{alice} as a *one-step neighbor* of G_{bob}, and consider a *k-step neighbor* of G_{bob} to be one created by adding or removing k individuals from G_{bob}. *Smooth sensitivity* is a technique which computes privatization noise based on both the local sensitivity of the true data-set, *and* the local sensitivity of all *k-step* neighbors scaled by k, for all k [7]. The technique 'smooths' over the local-sensitivity jumps depicted in the alice-bob graph example. However, local-sensitivity based techniques satisfy a weaker definition of differential privacy, and in some cases computing the amount of noise required to privatize a given D_I may be infeasible. We will primarily focus on techniques which satisfy strict ϵ-differential privacy in this chapter, but we will reference existing smooth-sensitivity techniques where applicable, and we recommend looking at [8] for more information on this approach.

3 Differential Privacy and Network Data

The above definition for differential privacy assumes all information about a data-set participant is provided by the participant themselves; protecting an individual's presence in the data-set then protects all the information regarding them. The situation changes when we ask survey participants to provide information about other individuals.

We will refer to individuals who contribute their knowledge to the data-set as *participants*, and individuals who have information provided *about* themselves (by others) as *subjects*. Traditional differential privacy protects participants only, and

in many cases it seems clear that subject privacy is unnecessary: if a survey counts the students who attended the "Coffee with the Dean" event, the dean's privacy is not important. By contrast, a study that counts students who report having sexual relations with the football captain exposes extremely sensitive information about its subject. Social networks are often collected from populations of interest by having participants list the full names of their friends within the population; these relationships form directed network edges leading from the participant's node to the nodes of each of their friends [9]. In this case, the friends are subjects of the participant's survey data, but the participant herself may also be the subject of some of her friends' survey data (if they also submit surveys). This presents a complex situation in which to apply differential privacy.

The core of the differential privacy guarantee is that the privatized result R is difficult to attribute to the true world versus one of its neighboring possible worlds. Adapting differential privacy to networked data amounts to deciding what we mean by "neighboring worlds" in this context. There are several possibilities; each one provides a different level of privacy guarantee and deals with a different type of "gap" between worlds. As always, there is a trade-off between privacy and utility: in general, the stronger the privacy guarantee, the more noise will be required to achieve it. We will describe two network privacy standards, *node privacy* and *edge privacy*, which have appeared in the literature.

Additionally, we propose two novel standards, *out-link privacy* and *partition-privacy*, that require less noise than existing standards; give reasonably strong guarantee of privacy similar to traditional differential privacy; and enable certain queries that required levels of noise that rendered results meaningless under existing standards.

3.1 Node Privacy

A privatized query Q satisfies *node-privacy* if it satisfies differential privacy for all pairs of graphs $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ where $V_2 = V_1 - x$ and $E_2 = E_1 - \{(v_1, v_2) | v_1 = x \vee v_2 = x\}$ for some $x \in V_1$.

The Alice-Bob graph example in Sect. 2 implicitly assumes this privacy standard: In node privacy, if the true world is a given social network G, the neighboring possible worlds are ones in which an arbitrary node, and *all* edges connected to it, are removed from or added to G. This privacy guarantee completely protects *all* individuals, both participants and subjects. An attacker in possession of R will not be able to determine whether a person x appears in the population at all. Although this is an natural adaptation of differential privacy to social networks, it also places *extremely* severe restrictions on the queries we are able to compute, as we will demonstrate in Sect. 4, and in many cases, node-privacy may be an unnecessarily strong guarantee.

3.2 Edge Privacy

A privatized query Q satisfies *edge-privacy* if it satisfies differential privacy for all pairs of graphs $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ where $V_1 = V_2$ and $E_2 = E_1 - E_x$ where $|E_x| = k$.

In edge privacy, if the true world is the social network G, neighboring possible worlds are ones in which k arbitrary edges are added or removed from G. An attacker in possession of R won't be able to determine with high certainty whether individuals x and y are friends, and an individual node in the graph can plausibly deny the existence of up to k of its friendships with other nodes. Single edge privacy, with $k = 1$, is the standard most often used in existing literature on differentially private graph analysis. This is a weaker guarantee than node-privacy: high-degree nodes may still have an easily identifiable effect on query results, even though their individual relationships are protected. However, this is a sufficiently strong for many applications, and enables many more types of queries to be privatized than the severely-restrictive node-privacy.

3.3 Out-Link Privacy

A privatized query Q satisfies *out-link privacy* if it satisfies differential privacy for all pairs of graphs $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ where $V_1 = V_2$ and $E_2 = E_1 - \{(v_1, v_2)|v_1 = x\}$ for some $x \in V_1$.

This privacy guarantee protects the data contributed by data-set *participants*, using the same conceptual privacy standard as the original definition of differential privacy. Given that the true world is a social network G, the neighboring possible worlds are ones in which an arbitrary node and all of its *out-links* are removed from or added to G. An attacker in possession of R won't be able to determine whether a person x supplied their data (submitted a survey) to help produce the graph. This privacy guarantee is strictly weaker than node privacy, but compares well with single edge privacy for many queries. Any participant can plausibly deny its out-links, or, equivalently, any participant can plausibly deny one in-link from another participant node. Analogous to k-edge privacy, we can also provide k-out-link privacy by considering neighboring worlds that differ from the true world by the out-links of up to k nodes. Note that 2-out-link privacy allows two nodes to *simultaneously* deny all out-links, and as a result, this enables a complete mutual edge to be protected (providing single-edge privacy in addition to out-link privacy). In general, a k-level privacy guarantee can be satisfied by scaling the added noise by k.

Out-link privacy improves on edge-privacy by reducing the distinctive signature of high-degree nodes in the data-results, through protecting all relationships cited *by* the popular person: although others may still claim to be friends with her, she can plausibly deny those relationships are mutual. Additionally this standard simplifies sensitivity computation and noise addition, enabling many queries that would be infeasible under both node and edge privacy as we will demonstrate in Sect. 4.

3.4 Partition Privacy

Define a partitioned graph to be comprised of separate components such that $G = \{g_i\}$ for disjoint subgraphs g_i. A privatized query Q satisfies *partition privacy* if it satisfies differential privacy for all pairs of graphs G_1, G_2 where $G_1 = G_2 - g_i$ for some $g_i \in G_1$.

Many questions about social structures are naturally asked over a collection of graphs rather than one monolithic social network. Social scientists studying interpersonal interaction run experiments over large collections of small social groups, collecting social networks for each distinct group [10, 11]. Collections of disjoint social networks can be implicit in larger graphs as well. Node properties such as *dormitory, major, university*, or *geographical location* can be used to partition large graphs into meaningful sets of disjoint local social networks [12]. This enables researchers to perform tests of hypotheses about social behavior across groups, such as "Is clustering coefficient correlated with gender in dormitory friendship structures?".

This useful sub-class of analyses is especially amenable to privatization. In partition privacy, neighboring possible worlds are ones in which one subgraph is added or removed from the set of disjoint subgraphs comprising the data-set. Partition privacy is strictly stronger than node-privacy: it provides protection at the level of entire social groups rather than individuals. However, it also requires very little noise to implement. We will present a diverse selection of analyses that can be easily privatized under partition privacy.

Below we will discuss the application of these four privacy standards to common social network analysis tasks such as triangle counts (and subgraph-counts generally), degree distributions, centrality measures, graph-modeling, and other differentially privatized network analyses from the existing literature. In addition to covering previous work, we provide several infeasibility proofs and propose two original algorithms applying out-link privacy to common problems in social network analysis.

4 Applications of Differential Privacy to Social Network Analysis

We now present a straightforward guide to the application of differential privacy to several common social network analysis techniques.

4.1 Triangle Counting

Triangles, instances in which two of an individual's friends are themselves mutual friends, indicate social cohesion in the network. Triangle counts are the key parameter in the clustering coefficient, a common metric for describing and comparing graphs.

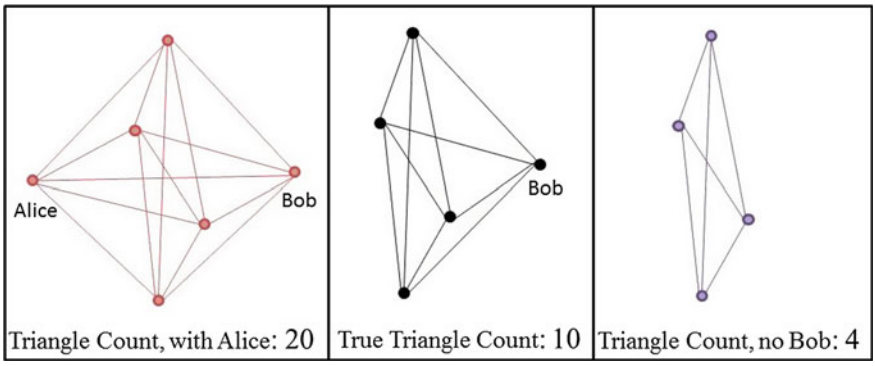


Fig. 2 Node-sensitivity of triangle-counts is a function of n, and thus is unbounded in general

Similarly, counts of other subgraphs such as stars, or squares, are used as graph statistics for graph similarity comparisons [13, 14]. All subgraph counts have similar privacy properties to the triangle count privatization described below.

Node Privacy Differentially private triangle counts are not feasible under simple node-privacy. In the worst case, adding a node to a complete graph of size n (a graph containing all possible edges), will introduce $\binom{n}{2}$ new triangles (Fig. 2). Since the change is dependent on the size of the graph, the global sensitivity of the query in general is unbounded: it is impossible to compute a finite global upper-bound (see Sect. 3).

Although the global sensitivity of the query is unbounded here, there is another approach, using ideas similar to the smooth sensitivity approach described in Sect. 2.1. If it is publicly known that the maximum degree of a graph is d, then removing or adding a node can affect the triangle count by at most $\binom{d}{2}$. And, any graph whose maximum degree is greater than d will have a k-step neighbor, for some k, whose maximum degree will be d (i.e., high-degree nodes can be removed until the maximum degree of the graph falls within the threshold). On generally sparse graphs with few nodes above degree d, the number of triangles in this bounded-degree neighbor graph will be a close approximation of the correct answer. The operation of finding the low-degree neighbor incurs its own sensitivity cost, but privacy can be still achieved at a sensitivity cost in the range $O(d^2)$ [15]. While this is untenable for large social networks, networks with low maximum degrees may successfully apply node-privacy to their triangle counts using this method.

Edge Privacy For similar reasons to node privacy, edge privacy is also not feasible for triangle-counts. In the worst case, removing an edge from a graph with n nodes can remove $n - 2$ triangles (Fig. 3). Since the sensitivity is a function of the graph size, it is unbounded in general.

However, the local sensitivity of this query under edge-privacy, the sensitivity over a specific data-set, is bounded. Consider two nodes, a and b, that have k wedges (paths of length 2) connecting them, as in Fig. 3. If G is a graph in which no pair of nodes has more than k wedges connecting them, then adding an edge to G will create

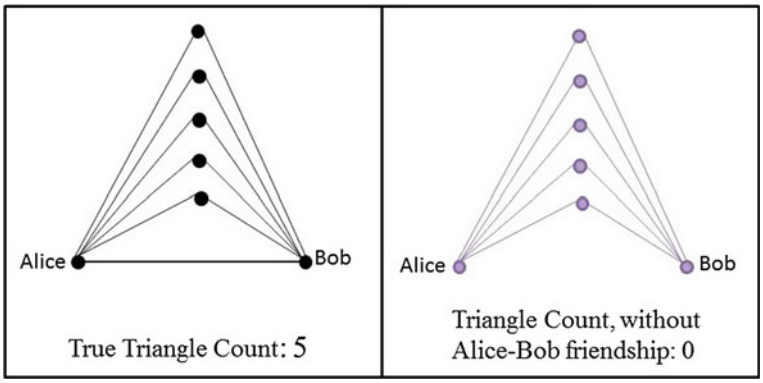


Fig. 3 Edge-sensitivity of triangle-counts is a function of n, and thus is unbounded in general

at most k triangles, and removing an edge will delete at most k triangles. We can apply smooth sensitivity techniques to take advantage of this in cases where k is not large, and thus attain a slightly weaker level of differential edge-privacy; however, real world social networks are transitive (if two people share a mutual friend, they're much more likely to be friends with each other) and this will tend to produce large values of k. When k is large, even instance-based noise addition may introduce error of a factor of 10 or greater [8].

Outlink Privacy We now propose a method for privatizing information about triangle counts and clustering coefficients under out-link privacy, using a somewhat modified version of the query that more closely mimics the information gathered from a real world social-network survey. To do this, we introduce a simple, powerful method that can be applied to gather private estimates of a variety of useful statistics over nodes in the graph.

By focusing on protecting the knowledge each individual has about their role with respect to the network, out-link privacy fits naturally with the techniques of *ego-network analysis*, an approach to social network analysis that focuses on the network as viewed by the individuals belonging to it [16]. In this approach, a network with n members is broken into n overlapping ego-network subgraphs, each consisting of a individual 'ego' node and his or her immediate neighborhood of friends (referred to as alters). A survey collecting information about the triangles in an individual's ego-network might look like Algorithm 1.

The only data that is retained by the researcher is, for each individual x: $outdegree(x)$, the number of friends the individual has, and $trianglecount(x)$, the number of triangles the individual participates in. These statistics are sufficient to determine the local clustering co-efficient of the node: the ratio between the number of triangles the node participates in and the maximum possible number of triangles for a node of that degree [13].

Out-degree and local clustering data from this survey can be collected into a two-dimensional histogram that provides detailed information about the patterns of social

Algorithm 1 A survey gathering information about triangles

function TRIANGLEQUERY
  $friendlist \leftarrow$ Query("Who are your friends?")
  $friendpairs \leftarrow$ CrossProduct($friendlist$, $friendlist$)
  $outdegree \leftarrow$ Size($friendlist$)

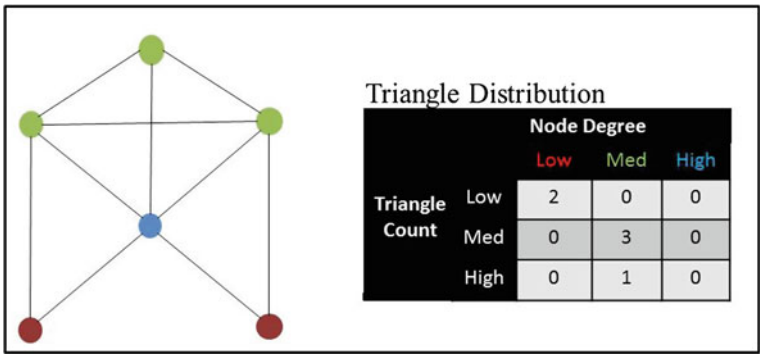
  $triangles \leftarrow$ Query("Which of these pairs are friends with each other?", $friendpairs$)
  $trianglecount \leftarrow$ Size($triangles$)
  return ($outdegree$, $trianglecount$)
end function

cohesion of the graph and has a very low sensitivity under out-link privacy: removing or adding an individual's survey data to the histogram only alters one partition count by at most one, and thus the noise required to privatize this data-structure would be very small. Histograms with fewer partitions and larger count values in each partition are less sensitive to added noise; we propose Algorithm 2 that produces a very flexible, robust, and safely privatized representation of the social cohesion patterns in the network using local triangle counts.

Algorithm 2 Privatizing local clustering coefficient distribution data

function PRIVATECLUSTERING(deg_{low}, deg_{med}, $data$)
  Initialize($bins$[][])
  for all ($nodeDegree$, $triangleCount$) $\in$ $data$ **do**
    $degBin \leftarrow$Partition($nodeDegree$, deg_{low}, deg_{med})
    $localCluster \leftarrow triangleCount/(nodeDegree * (nodeDegree - 1))$
    $triBin \leftarrow$Partition($localCluster$, 1/3, 2/3)
    $bin[degBin][triBin] \leftarrow bin[degBin][triBin] + 1$
  end for
  for $i = 0 \rightarrow 2$, $j = 0 \rightarrow 2$ **do**
    $bins[i][j] \leftarrow bins[i][j]+$ LaplacianNoise(1)
  end for
  return $bins$
end function

Algorithm 2 takes as input two node-degree threshold values, deg_{low}, deg_{med} and uses these to partition the ($outdegree$, $trianglecount$) data-points collected from the $TriangleQuery$ survey into low, medium and high degree nodes. The algorithm then computes the local clustering coefficient of each node and further partitions nodes by these values, creating a histogram with nine partitions (see Fig. 4). Laplacian noise sufficient to cover a function sensitivity of 1 is added to each partition, and the privatized result may be released. We can consider the effect of this noise in terms of how many of the noisy, privatized partition counts can be expected to differ measurably from their true values. With only nine counts and a sensitivity of 1, the expected number of privatized partition counts that will differ from their true values by more than 3, is less than 0.25. The released histogram accurately and succinctly



Triangle Distribution

Triangle Count	Node Degree: Low	Node Degree: Med	Node Degree: High
Low	2	0	0
Med	0	3	0
High	0	1	0

Fig. 4 The triangle distribution allows us to present clustering information with an out-link sensitivity of 1

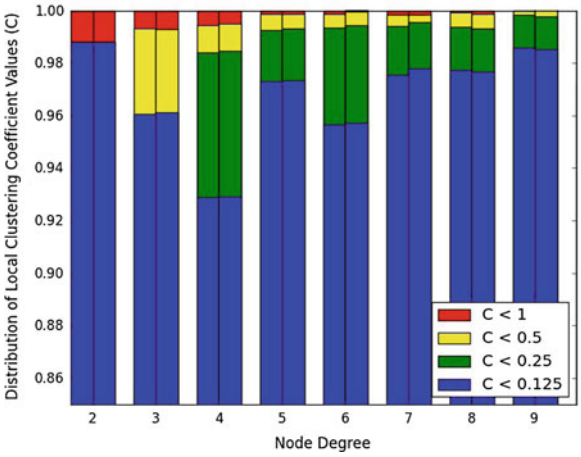


Fig. 5 A comparison of true and privatized results over the Slashdot Zoo social network

captures useful information about the distribution of meaningful local patterns across the graph.

Figure 5 shows the effect of privatization noise on the outlink-private triangle counting algorithm over the Slashdot 'Zoo' social network [17]. Here, nodes are binned by their degree and their local clustering coefficient, and the resulting degree-bins are normalized to show the distribution of clustering coefficients in each degree bin. The true counts are given in the left columns, their privatized versions are given in the right. Note that, as the magnitude of noise added by this algorithm is very small in comparison to the scale of the data, it was necessary to focus on a small section of the results in order for the effect of the noise to be visible.

The same simple approach can be used to collect and privatize any information available within an ego-network, simply by restructuring the survey appropriately. For example, replacing question 2 in the survey of 1 by the question "For each of

your friends, add a check mark if the two of you share at least one additional, mutual friend" will collect information about the probability that an edge participates in a triangle. The question "Are you part of a group of at least k friends who are all mutual friends with each other?" collects statistics about cliques in the graph.

If undirected social network data must be privatized, the survey-collection approach described above may be simulated by considering each node's immediate neighborhood as their ego-network view, and sub-sampling by introducing α probability that the ego is unaware of any given edge between its alters.

Partition Privacy In applications that require a collection of disjoint social networks, even more detailed privatized analysis is possible. Partition-privacy allows essentially arbitrary analysis of individual graphs in the data-set and then privatizes the aggregation of the independent results. Assume an analysis has been performed on each individual graph, producing either a numerical result with a publicly known range (e.g., the global clustering coefficient of the graph), a category result (the gender of the dorm represented by the graph), or any combination of numerical and categorical results. The collection of graphs may now be viewed as a collection of multi-attribute data points. Removing or adding one graph from the collection is equivalent to removing or adding one of these data points; we can apply traditional differential privacy techniques to this set of independent data points as though we were working with tabular data over individuals. Two low-sensitivity techniques are very useful here: histograms and privatized means. We will demonstrate the application of these techniques in the examples below, beginning with an application of partition privacy to triangle-count data.

The global clustering coefficient is the proportion of wedges in the graph (where one person has a pair of friends) that are closed to form a triangle (i.e., the pair of friends are also friends with each other); formally, $Clustering\ Coefficient(G) = \frac{3*[number\ of\ triangles\ in\ G]}{[number\ of\ wedges\ in\ G]}$. A graph with no triangles has a clustering coefficient of 0; a clique has a clustering coefficient of 1. The clustering coefficient of a graph is a useful normalized measure of its social cohesion. However, it is difficult to draw meaningful conclusions about the population being studied using one piece of data in isolation. Given a collection of social networks, we can identify meaningful patterns of behavior by comparing clustering coefficients across networks.

Assume we want to examine how attribute X of a social group affects its degree of social cohesion. For example, we could study the relationship between the gender of a college dormitory and the clustering coefficient of the social network within the dorm (see Fig. 6). Given a data-set consisting of a collection of social networks for each possible value of X (e.g., a set of male, female and co-ed dorms), we first compute the global clustering coefficient over each individual network. We can then compute the mean of the clustering coefficients for each value of the attribute X, add noise to privatize the result, and release the privatized means.

The mean of a set of bounded numerical values has low sensitivity when the number of values is publicly known. Consider the mean $MaleDormsClustering = M/N$ where $M = \Sigma_{G \in MaleDorms}\, clustering\_coefficient(G)$ and N is the number of male-only dorms in the data-set. If N is publicly known (for instance, because

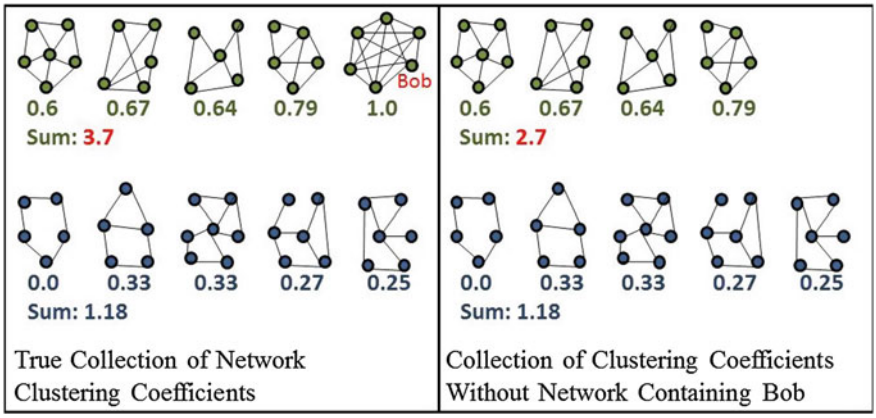


Fig. 6 Removing or altering one graph from the partitioned graph set only affects the numerator of the mean by one

each university's dorms are listed on their website) we can safely skip adding noise to this value and focus on privatizing only the numerator M without reducing the privacy of the result [18]. Since M is a sum of clustering coefficients that have values in the bounded range [0, 1], adding, removing or altering one clustering coefficient will alter the sum M by at most 1. Thus the sensitivity of the sum M is 1, and the value $\frac{M+Lap(1/\epsilon)}{N}$ will be differentially private. Note that the noise added to the true values of $MaleDormsClustering$ has a standard deviation of only $Lap(1/\epsilon)/N$.

4.2 Degree Distribution

The degree distribution of a graph is a histogram partitioning the nodes in the graph by their degree; it is often used to describe the underlying structure of social networks for purposes of developing graph models and making similarity comparisons between graphs [19].

Node Privacy Although degree distributions are represented as histograms, the sensitivity is not small under node privacy because one node affects multiple counts in the distribution: removing a node from the graph reduces the degree of all nodes connected to it. A node with k edges can affect a total of $2k + 1$ values of the distribution (Fig. 7). In the worst case, adding a node of maximal degree will change $2n + 1$ values, and since this sensitivity is dependent on n, it will be unbounded in general (see Sect. 3).

Edge Privacy Edge privacy is feasible for degree distributions. Removing one edge from the graph changes the degree of two nodes, and affects at most four counts (Fig. 8). Under k-edge privacy, the sensitivity is $4k$. With a sufficiently large graph,

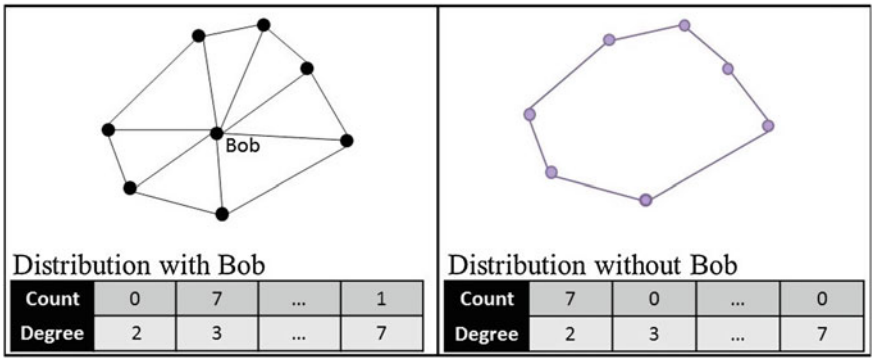


Fig. 7 Node sensitivity of degree distribution queries is a function of n, and thus is unbounded in general

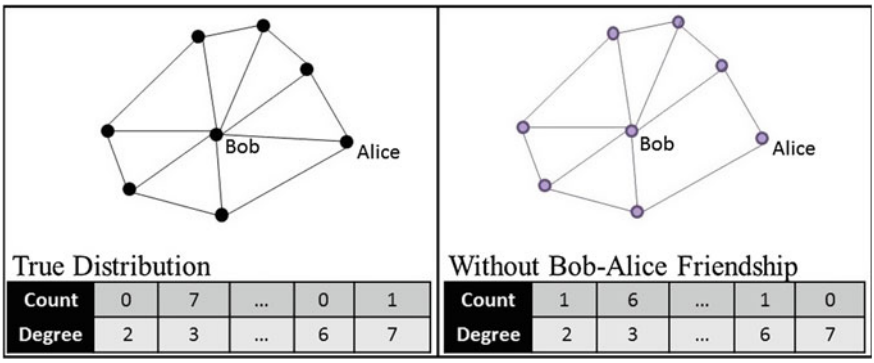


Fig. 8 Edge sensitivity of degree distribution queries is 4: at most four values can change by one when a node is added or removed

this is a negligible amount of noise, and the utility of this technique has been successfully demonstrated [6].

Outlink Privacy Out-link privacy, in contexts where it is deemed sufficient, requires even less noise for degree distributions. Here, we consider just the distribution of out-degrees, the result of asking participants, "How many friends do you have?" Removing one node and its out-links from the graph affects only one value in the degree distribution (Fig. 9). Under this privacy standard, a high-degree node may still leave evidence of its presence in the data-set through the out-degrees of its friends. However, there are many possible explanations for a slightly higher-than-expected degree among nodes in the graph: they may represent additional friendships among the nodes, or outside friendships with individuals who were non-participants in the survey. Exploiting this vulnerability to guess the presence of a high-degree node with any certainty would require an attacker to possess near complete information about the true social network.

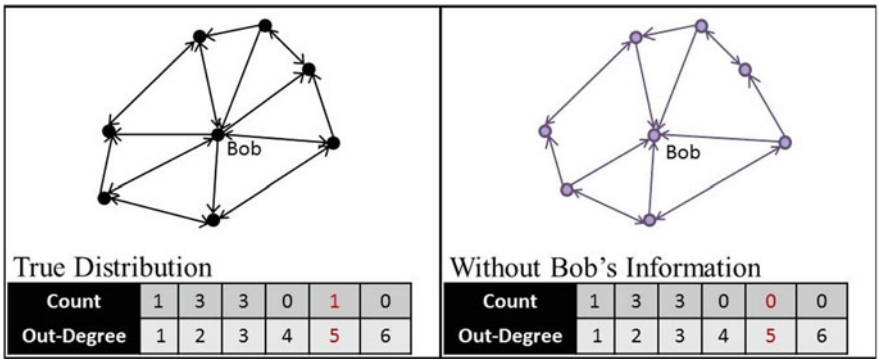


Fig. 9 Out-link sensitivity = 1. Protecting the out-edges of a node provides privacy with relatively little effect on the degree distribution

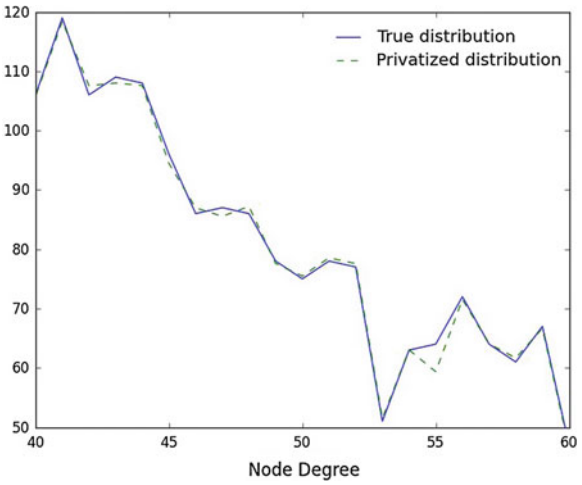


Fig. 10 A comparison of true and privatized results over the Slashdot Zoo social network

Figure 10 shows the effect of privatization noise on the outlink-private degree distribution algorithm over the Slashdot 'Zoo' social network [17]. Again, as the noise added by this algorithm is very small, the figure focuses on a subsection of the results in order to make the effect of the noise visible.

Partition Privacy Partition privacy can also enable privatized analysis of degree distribution data. Consider the context in which a researcher performs an experiment to directly study behavior patterns in small social groups. A common technique is to assign people to small groups where they must work cooperatively to solve problems [10, 11]. Interpersonal communications in each group are monitored and analyzed. Raw communication data can be transformed into social network graphs by adding edges between nodes that communicate frequently. In small groups, different degree

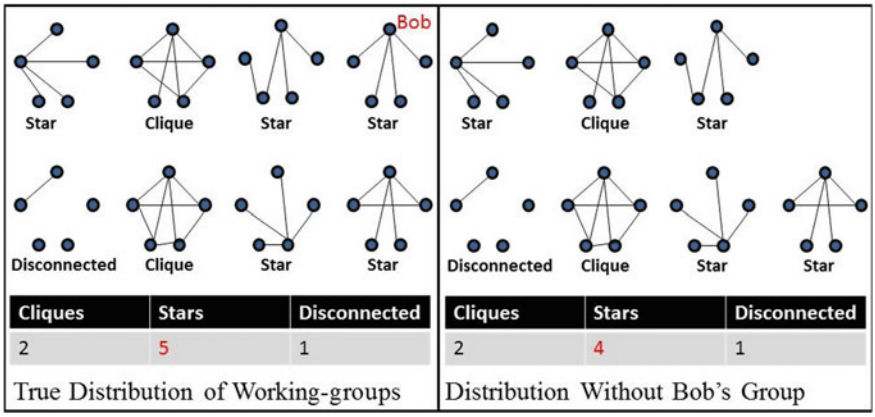


Fig. 11 Removing or adding one graph only affects the count in one histogram category by one

distributions will indicate different patterns of cooperation; for example, groups may have one high-degree 'leader' centralizing communication, or they might cooperate equitably together producing a near clique graph (see Fig. 11). These degree-distribution categories may be affected by the group's context (e.g., working in person, or online), and they may affect the group's performance on the assigned task. When degree-distributions help us attach a meaningful category label to individual networks, we can use a privatized histogram to safely release the distribution of these labels across the set of networks. If desired, we can further partition this histogram using properties such as the group's context or performance score to create more informative multi-dimensional histograms (for an example of a multi-dimensional histogram, see Fig. 14). As described in Sect. 2, histograms have a sensitivity of only 1 and may be safely released by adding Laplacian noise calibrated to that sensitivity to each count.

4.3 Centrality and Paths

Centrality measures attempt to gauge the relative "importance" of specific individuals within the social network; they may be studied on a per-node basis, identifying influential members of the community, or as distribution scores providing information about the overall behavior of the social network [20]. The simplest centrality measure is node degree: nodes with high degree are more likely to be influential in the network. However, other centrality measures take into account information from across the network: *betweenness* scores individuals by the number of shortest-paths between other pairs of nodes across the network that pass through them, and *closeness* scores nodes by the sum of their distances to all other nodes in the graph.

The two more complex centrality measures present difficulties for traditional approaches to differential privacy in social networks. Clearly, it is impossible to release a named list of influential individuals under node-privacy. But even distrib-

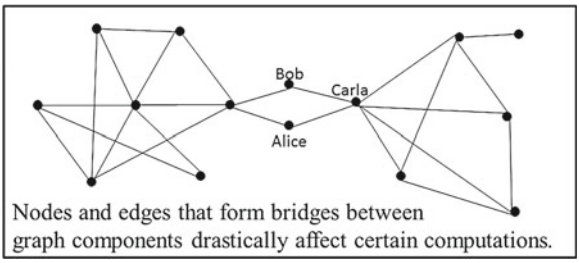


Fig. 12 Removing one node or edge from a graph can change path lengths catastrophically

utions of centrality scores can be very sensitive, under both node and edge privacy, due to the role of bridges in the graph. Removing a node, or edge, that forms the only connection between two otherwise disconnected subgraphs will have a catastrophic affect on path distances in the network, causing finite distances to become infinite, and thus will drastically alter betweenness and closeness scores (see Fig. 12). In general, privatizing traditional centrality measures, or any metric that relies on path lengths, remains an open problem under differential privacy.

Outlink Privacy We propose a very different approach for collecting and privatizing information about influential nodes within a network; one that satisfies out-link privacy (by protecting individuals' data contributions) and leverages individuals' knowledge about their community. We define a *popularity graph*: a synthetic network that represents the social structure among influential community members (Algorithm 3).

Individuals in the population are asked to "list up to three of your most popular friends within the specified population group". A base graph is created containing nodes for all members of the population group, and undirected edges of weight 0 are added between all pairs of nodes. The data collected from the survey is then added to the graph: when two popular people are listed on the same survey, the weight of the edge connecting them is incremented. For example, if a person submits a survey listing three popular friends, weights of every edge in the triangle connecting those friends will be incremented. The sensitivity of the popularity graph is 3, since a maximum of 3 edge-weight values can change if a participant adds or retracts their data.

To privatize the data, appropriate Laplacian noise to cover a function sensitivity of 3 is added to all edge-weights. Then two post-processing steps are applied: edges with low weight are eliminated, and the graph is anonymized. The resulting weighted popularity graph is published (Fig. 13). This graph can be used to understand the underlying social influence structure of the population, identifying social clusters and the bridges between them. The privacy of data provided by the query participants is fully protected; however, the subjects who appear as nodes in the graph will clearly be less secure and this analysis may not be appropriate in all contexts. For many population though, the popularity graph should be sufficient protection: anonymity, noisy edges, and the fact that the artificially-constructed graph will lack detailed substructures often used for

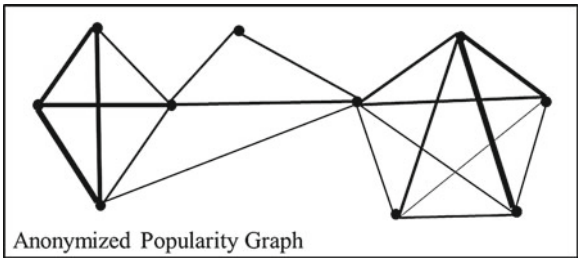


Fig. 13 A popularity graph with edge thickness indicating edge-weight

re-identification attacks, will all contribute to protecting the privacy of the query subjects.

Algorithm 3 Privatizing centrality data

```
function PRIVATECENTRALITY(importanceT, data_I)
    V ← population
    E[i][j] ← 0 ∀i, j ∈ V
    for all i ∈ I do
        ∀p_j, p_k∈data_I[i], E[p_j, p_k]←E[p_j, p_k] + 1
    end for
    for all i, j ∈ population do
        E[i, j] ← E[i, j]+ LaplacianNoise(3)
        if E[i, j] < importanceT then
            E[i, j] ← 0
        end if
    end for
    return PopularityGraph = (V, E)
end function
```

Partition Privacy A noteworthy property of partition privacy is that it does not exhibit the high sensitivity to path length queries that constrains other forms of graph privacy. Although removing a bridge will drastically affect path lengths in a given network, it will only affect *one* network in the collection of small disjoint networks that comprises the data-set for a partition privacy application. This enables privatized analysis for a wide variety of graph properties that are otherwise too revealing to be released.

The average shortest-path distance for a network is a measure of its connectedness. Given a collection of networks, we can find the average shortest-path length for each network and aggregate the results into a histogram, giving us information about the patterns of graph-connectedness across our data-set (see Fig. 14). As the sensitivity of a histogram is just 1, the results can be privatized by adding a relatively small amount of noise to each count. The same technique can be used on any numerical or categorical graph property: we can privatize the distribution of maximum centrality

Fig. 14 With a set of graphs, histograms can be used to release information about the relationships between multiple variables, including path lengths, with low sensitivity

scores, number of bridges per graph, or even graph diameters. This flexibility of application is one of the primary advantages of partition privacy.

4.4 Graph-Modeling and Social Recommendations

Several groups have proposed differentially private approaches to creating graph models–randomized synthetic graphs that are generated to be similar to a true, private, social network and thus can be studied safely in place of the original graph. The Stochastic Kronecker graph model has been privatized under edge-privacy [21], and several other groups have developed their own models that satisfy differential edge privacy [22–24].

We also note that the results from our proposed out-link privatized degree distribution and triangle statistics (see Sects 4.1, 4.2) could provide privatized input for the Transitive Chung Lu graph model proposed by Pfeiffer et al. [25]. This model is somewhat unique in the literature for its ability to generate graphs that match both the degree distribution and clustering coefficient of the original target graph.

Finally, the possibilities and difficulties of applying edge-privacy standards to social network recommendation systems are explored in [26].

5 Conclusions

Differential privacy represents a potentially powerful tool for analyzing social networks while providing strong guarantees of privacy for individual participants. The application of differential-privacy guarantees to social-network analysis allows results to be released with confidence that individual data will not be compromised by malicious attackers, even with the benefit of arbitrary background knowledge.

By providing this guide to differentially private social network analysis, along with new, powerful techniques for privatizing social-network data, we hope to spur the application of these standards to social-network data in a practical fashion. In future work we plan to study the application of out-link privacy and partition privacy to other social-network analysis tasks and provide studies of these approaches on real-world network data.

Acknowledgments This work was supported by the Center for the Science of Information, an NSF Science and Technology Center.

References

1. Narayanan A, Shmatikov V (2008) Robust de-anonymization of large sparse datasets. In: Proceedings of the 2008 IEEE symposium on security and privacy, pp 111–125
2. Zheleva E, Getoor L (2011) Privacy in social networks: a survey. In: Aggarwal CC (ed) Social network data analytics, p 277
3. Narayanan A, Shmatikov V (2009) De-anonymizing social networks. In: 2009 30th IEEE symposium on security and privacy, pp 173–187
4. Dwork C, McSherry F, Nissim K, Smith A (2006) Calibrating noise to sensitivity in private data analysis. In: Proceedings of the 3rd theory of cryptographyconference. pp 265–284
5. Hay M, Rastogi V, Miklau G, Suciu D (2010) Boosting the accuracy of differentially private histograms through consistency. Proc VLDB Endow 3(1–2):1021–1032
6. Hay M, Li C, Miklau G, Jensen D (2009) Accurate estimation of the degree distribution of private networks. In: IEEE international conference on data mining, pp 169–178
7. Nissim K, Raskhodnikova S, Smith A (2007) Smooth sensitivity and sampling in private data analysis. In: Proceedings of the thirty-ninth annualACM symposium on Theory of computing. ACM
8. Karwa V, Raskhodnikova S, Smith A, Yaroslavtsev G (2011) Private analysis of graph structure. In: Proceedings of the VLDB Endowment, vol 4(11)
9. Marsden P (1990) Network data and measurement. Annu Rev Sociol 435–463
10. Sparrowe RT, Liden RC, Wayne SJ et al (2001) Social networks and the performance of individuals and groups. Acad Manage J 44:316–325
11. Gladstein DL, Reilly NP (1985) Group decision-making under threat-the tycoon game. Acad Manage J 28:613–627
12. Traud AL, Mucha PJ, Porter MA (2011) Social structure of facebook networks. Physica A 391:4165–4180
13. Watts DJ, Strogatz SH (1998) Collective dynamics of "small-world" networks. Nature 393(6684):440–442
14. Holland P, Leinhardt S (1976) Local structure in social networks. Sociol Method 7(1)
15. Blocki J, Blum A, Datta A, Sheffet O (2012) Differentially private data analysis of social networks via restricted sensitivity. CoRR abs/1208.4586

16. Marin A, Wellman B (2010) Social network analysis: an introduction. In: Handbook of social network analysis, p 22
17. Leskovec J, Lang KJ, Dasgupta A, Mahoney MW (2008) Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters. CoRR abs/0810.1355
18. Christine Task CC, Publicly constrained populations in differential privacy
19. Newman M (2003) The structure and function of complex networks. SIAM Rev 167–256
20. Degenne A, Forsé M (1999) Introducing social networks. SAGE Publications Ltd, New York
21. Mir DJ, Wright RN (2009) A differentially private graph estimator. In: Proceedings of the 2009 IEEE international conference on data mining workshops. IEEE Computer Society, pp 122–129
22. Proserpio D, Goldberg S, McSherry F (2012) A workflow for differentially-private graph synthesis
23. Sala A, Zhao X, Wilso C, Zheng H, Zhao BY (2011) Sharing graphs using differentially private graph models. In: Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference, New York, NY, USA, ACM, pp 81–98
24. Gupta A, Roth A, Ullman J (2012) Iterative constructions and private data release. In: TCC, pp 339–356
25. Pfeiffer III PP, Fond TL, Moreno S, Neville J (2012) Fast generation of large scale social networks with clustering. CoRR
26. Machanavajjhala A, Korolova A, Sarma AD (2011) Personalized social recommendations: accurate or private. Proc VLDB Endow 4(7):440–450

Complex Network Analysis of Research Funding: A Case Study of NSF Grants

Hakan Kardes, Abdullah Sevincer, Mehmet Hadi Gunes and Murat Yuksel

Abstract Funding from the government agencies has been the driving force for the research and educational institutions particularly in the United States. The government funds billions of dollars every year to lead research initiatives that will shape the future. In this chapter, we analyze the funds distributed by the United States National Science Foundation (NSF), a major source of academic research funding, to understand the collaboration patterns among researchers and institutions. Using complex network analysis, we interpret the collaboration patterns at researcher, institution, and state levels by constructing the corresponding networks based on the number of grants collaborated at different time frames. Additionally, we analyze these networks for small, medium, and large projects in order to observe collaboration at different funding levels. We further analyze the directorates to identify the differences in collaboration trends between disciplines. Sample networks can be found at http://www.cse.unr.edu/~mgunes/NSFCollaborationNetworks/.

Keywords Complex networks · Complex network analysis · Research funding networks · Six degrees of separation · NSF

H. Kardes
inome Inc, 500 108th Ave NE, Bellevue, WA 98005, USA
e-mail: hkardes@cse.unr.edu

A. Sevincer
Intel Corporation, 2501 NW 229th Ave, Hillsboro, OR 97124, USA
e-mail: asev@cse.unr.edu

M. H. Gunes(✉) · M. Yuksel
University of Nevada, Reno, 1664 N Virginia Street, Reno, NV 89557, USA
e-mail: mgunes@unr.edu

M. Yuksel
e-mail: yuksem@cse.unr.edu

F. Can et al. (eds.), *State of the Art Applications of Social Network Analysis*, Lecture Notes in Social Networks, DOI: 10.1007/978-3-319-05912-9_8,

1 Introduction

As data about social networks has grown vastly in size and heterogeneity, complex network analysis of such networks have become more popular. Many researchers are modeling the growth and the structure of the networks from different fields including biology, chemistry, geography, mathematics and physics. Complex network analysis helps to capture the small-scale and the large-scale features of these networks that are not evident. Such analysis may uncover the underlying dynamics of the network patterns. In this direction, researchers have investigated interactions of different systems in various fields as a complex network [1].

Many studies [2–5] look into popular social networks such as Facebook, Twitter and YouTube. Newman provided the first study on co-authorship networks by analyzing the macroscopic properties of different domains [6, 7]. Similarly, researchers have studied academic ties [8], air transport [9], authors network [10], citation networks [11, 12], friend recommendation [13], influenza spread [14, 15], Internet topology [16–18], news networks [19, 20], patent networks [21, 22], protein interactions [23], software collaborations [24, 25], and video industry [26] as complex networks.

In this chapter, we analyze the collaboration of researchers when they obtain federal funding.[1] For this study, we obtain the funding data of the National Science Foundation (NSF), an independent federal agency established by the U.S. Congress in 1950 to promote the progress of science; to advance the national health, prosperity, and welfare; and to secure the national defense. NSF has an annual budget of about $7.4 billion (FY 2011) [28], and funds research and educational activities at various institutions including universities, research institutes, foundations and industry.

As a public institution, NSF shares its funding information [29]. The data released by NSF includes the Principle Investigator (PI), i.e., the researcher responsible for leading the project, co-PIs (if any), organizations, directorate, grant amount and several other fields for the funded projects. In order to analyze the collaboration structures within the NSF research funding network, we generate three types of networks from the provided dataset based on the number of collaborations for different time frames. First, we construct the PI collaboration network where we analyze the social interaction of researchers. The PI network shows the structure of the collaboration and different characteristics of the NSF grants among PIs. Moreover, from the institution information of co-PIs, we build an organization network where we inspect the collaboration among research institutions. This analysis reveals the most central organizations and collaboration trends. We also derive the state collaboration network to study the collaboration among the states in obtaining federal funding.

Since, we construct these networks both for different time frames and as a whole; we compare the network characteristics of these networks for different time frames and capture the changes in the NSF collaboration network over the time. Additionally, we analyze these networks for small, medium, and large projects in order to observe

[1] An earlier version of this study appeared in [27].

collaboration patterns at different funding levels. We further analyze the funding networks within each NSF directorate and find their distinct properties. We compare each directorate with the other directorates to better understand the collaboration in the NSF funding data.

The main goal of this chapter is to collect the NSF funding dataset, discover interesting complex network structures from the dataset, and derive new insights from it. The newly discovered properties from the dataset will give an idea of the collaboration among researchers in obtaining federal funding. Researchers have studied National Institutes of Health (NIH) and NSF data sets for visualization. For instance, Herr et al. presents an interactive two dimensional visualization of the 60,568 grants funded by NIH in 2007 [30]. However, this chapter is, to best of our knowledge, the first study to analyze the funding data as a complex network.

In the rest of the chapter, first we clarify the metrics that we use during our analysis and we describe data collection and network construction procedures. We then present analysis of research funding networks derived from the NSF data at different levels. Finally, we conclude and provide future directions.

2 Preliminaries and Definitions

There are several well known metrics which are widely utilized in complex network analysis. In this section, we briefly provide an overview of the metrics that we use in our analysis.

Size is one of the most basic properties of a network, and is quantified by the number of nodes n and the number of edges e.

The basic characteristic to infer a network's connectivity is **average node degree** $\bar{k} = 2n/e$. The **degree** k of a node is the number of edges that are adjacent to the node. A node with degree k is called as $k\text{-}degree$ node, and $n(k)$ is the set of all k-degree nodes in a network. The average node degree can also be calculated by taking the mean of the degree of all nodes in the network. **Weighted Degree** of a node is the sum of the weights of all of the edges that this node has. **Node degree distribution** is the probability distribution of the node degrees where the probability of having a $k\text{-}degree$ node in the network is expressed as $P(k) = n(k)/n$.

Distance is the shortest path length between a pair of nodes in the network. **Average Path Length** stands for the average distance between all pairs of nodes in the network. **Diameter** is the maximal shortest distance between all pairs of nodes in the graph, and gives an idea of how far apart are the two most distant nodes.

Assortativity illustrates the link behavior of nodes, and measures whether similar degree nodes are more likely to be connected to each other. **Rich Club** measures how well the highest degree nodes in the network are connected.

Clustering coefficient is the measure of how well the adjacency (i.e., neighbors) of a node are connected. The neighbor set ns of a node a is the set of nodes that are connected to a. If every node in the ns is connected to each other, then the ns of a is complete and will have a clustering coefficient of 1. If no nodes in the ns of a are

connected, then the clustering coefficient of a will be 0. High clustering coefficient is the indicator of **small-world** effect along with small average shortest path.

There are several measures for the *centrality* of a node within the network. Such centrality measures are important in analyzing the funding network since they may determine the relative importance of a node within the network. **Betweenness Centrality** of a node is the measure of how often this node appears on the shortest paths between any node pair in the network. **Closeness Centrality** of a node is the average distance of this node to all other nodes in the network. **Eigenvector Centrality** measures the importance of a given node based on its connections.

3 Data Collection

NSF provides historic information on funded grants at its website. A search engine provides access to the grant information. Each search query turns at most 3,000 grants at a time, and there is a rate limit for queries from a computer. This rate limiting of NSF website necessitates using multiple computers if one wants to download the grant data faster. We implemented a crawler using the PlanetLab [31] infrastructure to download the NSF grants database in order to parallelize the download process. Overall, we downloaded a total of 279,862 funded grant data spanning from 1976 to December 2011.

Each NSF grant has a Principal Investigator (PI), organization, co-PIs, directory and several other fields in the database. We ignored some of these fields since our aim is to analyze the network of collaborations among the NSF grants. The individual grants such as fellowships or presidential awards are not included in the dataset as they are not collaborative works. A collaborative research grant with co-PIs from the same institution has a single entity in the NSF database. However, if the co-PIs are from different organizations, there may be multiple entities in the database for this grant. If it appears in multiple entities, the title of the grant should be the same and begin with 'Collaborative Research'. We filter the dataset considering these rules and similar naming conventions of the NSF.

4 Networks Analysis of the NSF Funding

In order to analyze the collaboration patterns within the research funding network, we generated three types of networks from the dataset and visualized them with Gephi [32]. First network we explore is the *PI network*, i.e., the collaboration network between Principal Investigators (PIs) of the grants. By constructing this network, we aim to understand the relationships and characteristics of the collaboration between researchers. To construct the PI network, we connected co-PIs of each grant as in Fig. 1. In this network, each node $P_i \in PIs$ represents a PI and each edge between P_i and P_j indicates that these two PIs have a collaborative grant. This network is

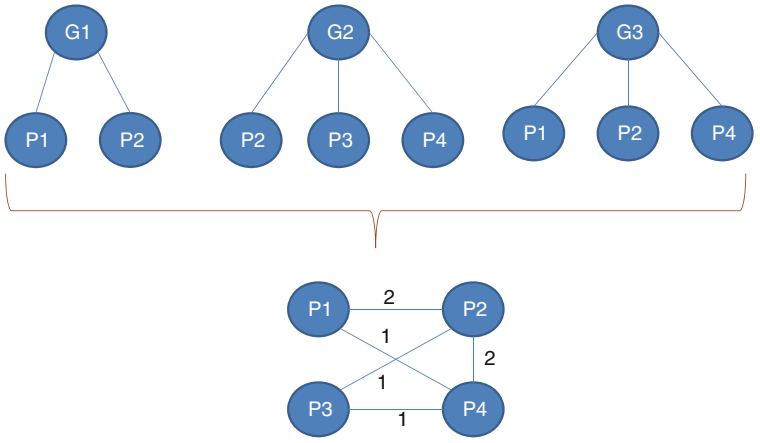


Fig. 1 PI network construction

weighted and the weight of the edges represents the number of grants collaborated among the two PIs. Moreover, we built the *organization network*, i.e., the collaboration network between the organizations of the PIs of the funded grants to observe the relations between institutions in receiving grants from the NSF. Finally, we constructed the *state network*, i.e., the collaboration network between the states of the PIs in order to analyze the patterns among the home state of researchers.

Furthermore, we drew the same networks for different time frames, namely 80s (i.e., 1980–1989), 90s (i.e., 1990–1999), and 2000s (i.e., 2000–2009). Although NSF was established in 1950, it has begun to gain more importance since 1980s as the country realized the significance of research in science, technology, and education. In 1983, NSF budget exceeded $1 billion for the first time. Later, it passed $2 billion in 1990, $4 billion in 2001 and became $6.9 billion in 2010. Therefore, in this analysis, we examine the evolution of the collaboration networks and the effect of the growth of the budget to the collaboration of the researchers and organizations.

Moreover, we analyzed these networks for small, medium, and large projects in order to observe the collaboration at different funding levels. Similarly, we analyzed the funding network within each NSF directorate to find their distinct properties. We compared each directorate with the other directorates to better understand the collaboration patterns within different research fields.

4.1 PI Network

The PI network constructed from the dataset is shown in Fig. 2. In this network, there are about 106K nodes and 197K edges which makes it hard to visualize. The diameter of the PI network, which is constructed from all PIs with a collaboration, is

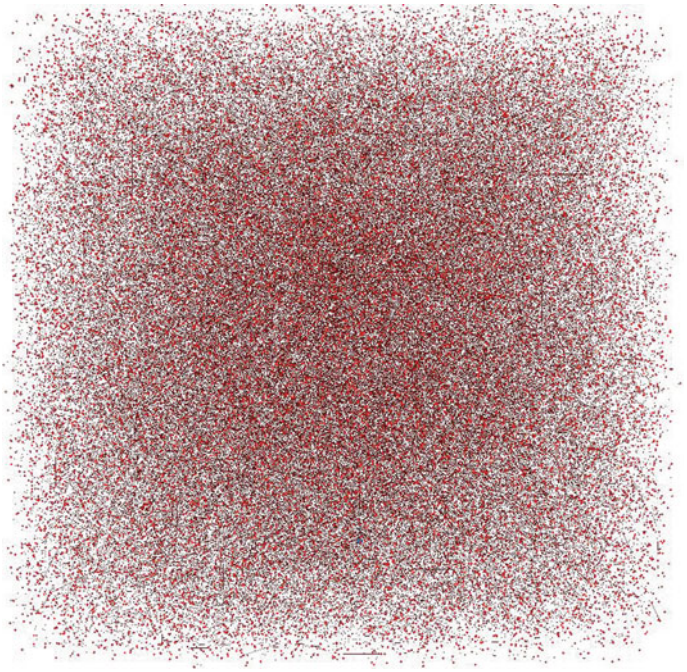


Fig. 2 PI collaboration network

29 and the average path length is 24.4. The average path length is higher than other similar social networks. There are several directorates such as Biological Sciences (BIO), Computer and Information Sciences (CSE), Education and Human Resources (EHR), Geosciences (GEO), Mathematical and Physical Sciences (MPS), Office of Polar Programs (OPP), Social Behavioral and Economic Sciences (SBE) in NSF grants. Thus, in our opinion, the main reason for having high diameter and average path length values for the PI network is due to the diverse fields of studies of the PIs. Additionally, as the PI network is sparse, the number of interdisciplinary grants which would make the PI network more connected is low. As indicated in the Directorates Networks Section, the PI network of each individual directorate is well-connected with low diameter and average path length values but we do not observe this behavior when we consider all of the directorates together.

Figure 3a presents the *clustering coefficient distribution* of the nodes in the PI network. The average clustering coefficient of the graph is 0.46. This is considerably higher than a random network of similar size, which happens in *small world* [33] networks.

The *node degree distribution* in Fig. 3b does not exhibit a power-law distribution as observed in many social networks but rather results in a declining curve. We think this is mainly due to the fact that funding collaborations require considerable effort and researchers are limited in the number of collaborations they can form. The

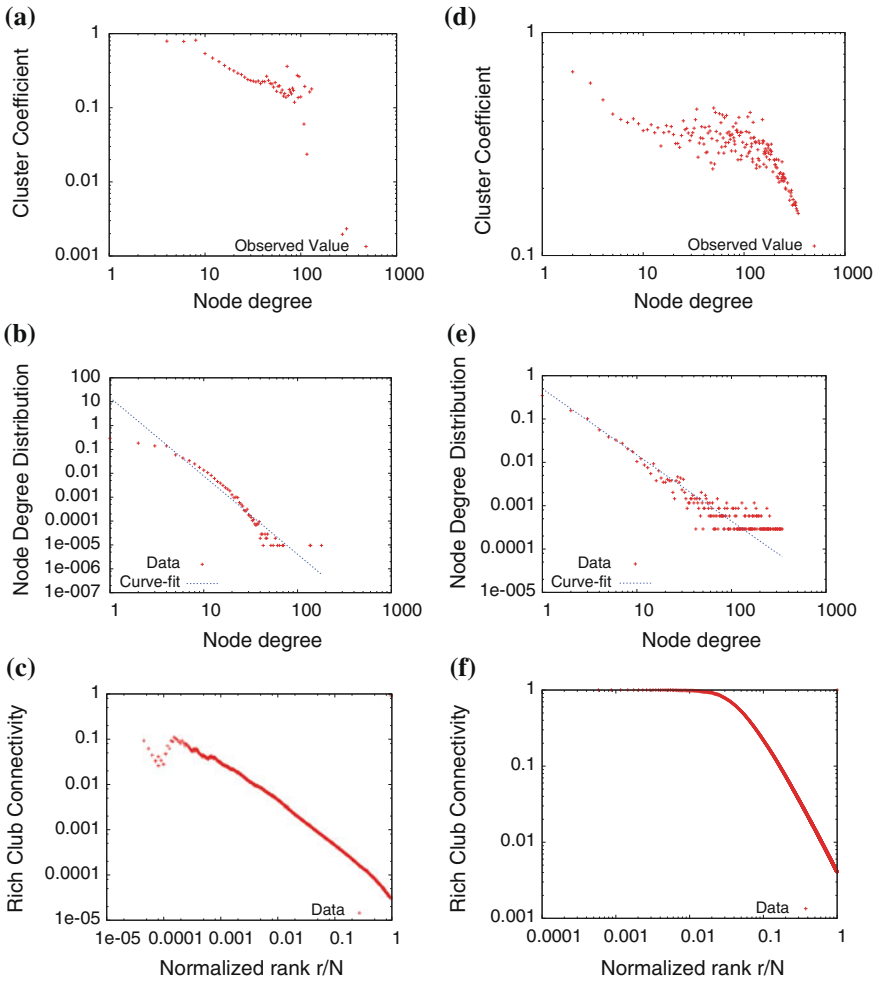


Fig. 3 PI and organization network metrics. **a** PI clustering coefficient. **b** PI degree distribution. **c** PI rich club. **d** Organization clustering coefficient. **e** Organization degree distribution. **f** Organization rich club

average node degree for the network is 3.71, while the *weighted node degree* is 4.5. The number of collaborations, if any, among PIs is 1.22 on average.

The *assortativity* of the graph is 0.18, which means the network is non-assortative [34]. That is, PIs who have high collaborations slightly tend to work together rather than collaborating with PIs that have low collaborations.

Moreover, Fig. 3c shows the *rich club connectivity* of the PI network. According to this graph, there is not an obvious rich club that contains most of the collaborations even though such phenomenon has been observed in citation networks [35].

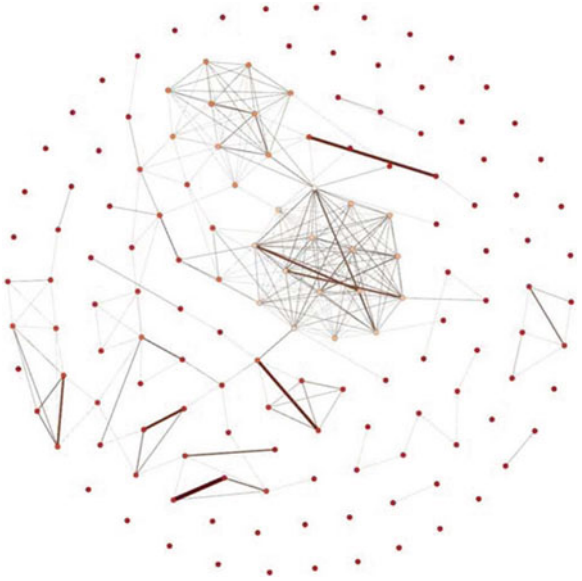


Fig. 4 PI collaboration network for PIs with high degrees

In order to better analyze highly collaborative PIs, we draw the network of the PIs with the highest node degrees in Fig. 4. In this figure, the thickness of the edges illustrate the number of collaborations among PIs while the boldness of the color of each node represents the weighted node degree, i.e., the total number of collaborative grants for that node. In the figure, we observe few cliques indicating a highly collaborative group of researchers and some isolated nodes indicating researchers with a large number of distinct collaborations.

Moreover, in order to study frequent collaborations among researchers, we construct the PI network by only considering the highest weighted edges in Fig. 5. As seen in the figure, there are many distinct pairs of PIs while there are a few triangles and larger cliques in this network. This indicates most of the frequently funded research teams consist of two PIs. Though more statistical evidence is needed, one may concur that frequent collaboration with another PI is more likely to increase chances of a successful project compared to new collaborations that might be more fruitful while being more risky.

4.2 Organization Network

To observe the relations between institutions receiving grants from the NSF, we build the *organization network*, i.e., the collaboration network between the organizations of the PIs of the funded grants. The constructed network of 3,450 nodes and around

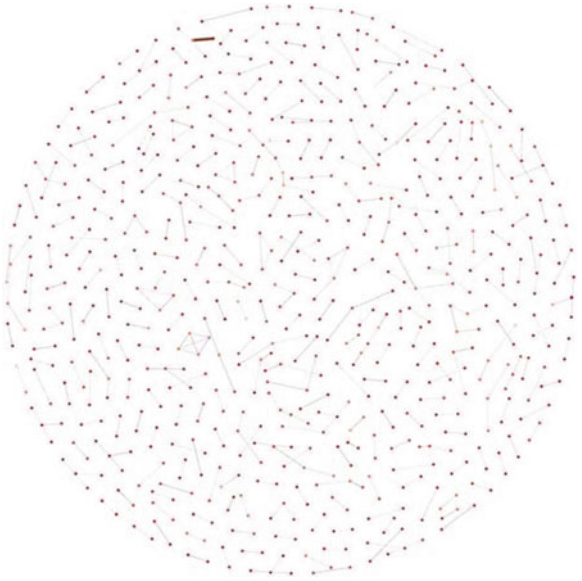


Fig. 5 PI frequent collaboration network

27K edges is visualized in Fig. 6. In this visualization, the nodes with high node degrees are located at the core of the network. The edge weights of these core nodes are usually high as well. This network is also weighted and the weight of the edges represents the number of grants collaborated among the two organizations. As seen in the figure, there is a group of nodes that are highly collaborative at the center of the figure.

The diameter of the organization network is 6.5 and the average path length is 3.07. However, we observed that there are many organizations that collaborate just once or twice. Many of these organizations are some short-run companies which were in business for a limited time. When we exclude such organizations from the network, the diameter of the network becomes 6.0 and the average shortest path becomes 2.75. Therefore, it can be concluded that the *six degrees of separation* is also observed in this network.

Figure 3d presents the *clustering coefficient distribution* of the nodes in the organization network. The average clustering coefficient of the network is 0.34. The top clique size is 20 indicating that there are 20 organizations that have pairwise collaborated with each other. Along with the small average path length, the very high clustering coefficient compared to a random network of similar size indicates the *small world* characteristics for the collaboration network of organizations.

The *node degree distribution* of the organizations network is shown in Fig. 3e. The degree distribution follows a power-law distribution with a fat tail. The average node degree for the network is 15.85, while the average weighted degree is 33.36. This indicates that on average each organization collaborated with its peers twice.

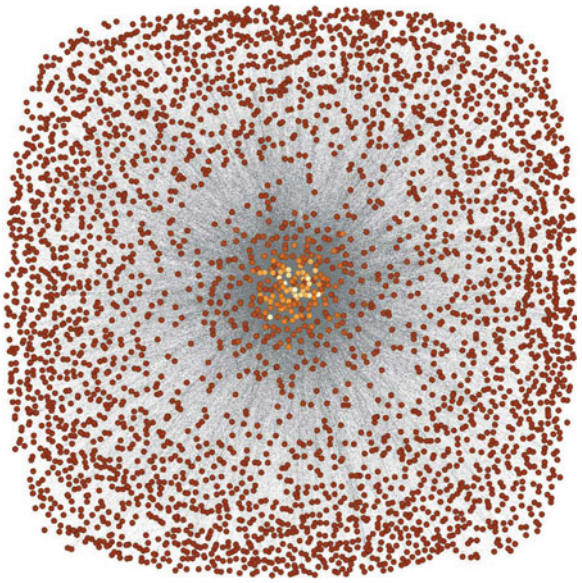


Fig. 6 Organization collaboration network

According to Fig. 3f which presents the *rich club connectivity*, there is a rich club among organizations that receive federal funding. As observed as a highly connected core in Fig. 6, a group of organizations participate in most of the collaborations. To further investigate the rich club, we calculate the betweenness centrality, node degree, and weighted node degree for each node. Table 1 shows the rankings of the top 10 organizations based on the betweenness centrality and node degree values. Essentially, these top 10 organizations are part of the rich club in the network. As indicated above, for an organization, node degree expresses the number of distinct organizations which a collaboration was made while weighted node degree represents the total number of grants collaborated with the other institutions. According to the table, University of Colorado at Boulder is ranked 1st both according to the betweennes centrality and node degree, while ranked 5th based on weighted degree. This illustrates that even though University of Colorado at Boulder has collaborated with the highest number of organizations, it is not the highest according to the total number of grants collaborated. Another interesting result is that even though MIT is not one of the top ten organizations based on the node degree, it is the 4th institution according to weighted node degree.

The *assortativity* value of this network is -0.09, which indicates that the organizations equally prefer to collaborate with high or low degree organizations. That is, different from the PI network where highly collaborating researchers slightly prefer to collaborate with researchers that also have high degrees, organizations are indifferent to the degree or popularity of the collaborators.

Table 1 Top 10 organizations

Metric	Organization rankings							
	1980s		1990s		2000s		Overall	
	Organization	Value	Organization	Value	Organization	Value	Organization	Value
Betweenness centrality	U. of Washington	11,540	U. of Washington	60,317	U. of Washington	47,912	U. of Colorado at Boulder	213,721
	U. of California-San Diego	11,147	U. of Minnesota-Twin C.	47,824	Arizona State U.	44,406	Arizona State U.	192,345
	U. of Michigan Ann Arbor	9,200	Pennsylvania State U.	43,756	U. of Colorado at Boulder	41,063	U. of Michigan Ann Arbor	183,380
	MIT	7,737	U. of Michigan Ann Arbor	37,827	Pennsylvania State U.	39,131	U. of Wisconsin Madison	182,452
	U. of Illinois at U-C	6,493	U. of Colorado at Boulder	37,214	GA Inst. of Technology	38,971	Pennsylvania State U.	180,111
	U. of California-Berkeley	6,358	U. of Wisconsin-Madison	36,730	U. of Michigan Ann Arbor	36,798	U. of Illinois at U-C	179,725
	Cornell U.	6,299	U. of California-Berkeley	35,989	Virginia Poly. Ins.	36,119	U. of Washington	175,303
	Colombia U.	6,275	Colombia U.	35,595	Ohio State U.	35,710	Colombia U.	163187
	Carnegie-Mellon U.	5,847	U. of Illinois at U-C	34,931	Colombia U.	34,570	MIT	153,406
	Indiana U. Of Pennsylvania	4,595	Suny at Stony Brook	33,796	U. of Minnesota-Twin C.	33,117	Cornell U.	151,373
Node degree	U. of Washington	52	U. of Washington	121	U. of Washington	225	U. of Colorado at Boulder	344
	U. of California-San Diego	49	U. of Minnesota-Twin C.	112	Pennsylvania State U.	212	U. of Washington	336
	U. of Michigan Ann Arbor	44	Pennsylvania State U.	101	U. of Colorado at Boulder	205	U. of Wisconsin-Madison	330
	MIT	43	U. of Illinois at U-C	101	U. of Illinois at U-C	200	Colombia U.	324
	U. of Illinois at U-C	43	U. of Colorado at Boulder	100	U. of Arizona	197	Pennsylvania State U.	323
	U. of California-Berkeley	41	U. of Wisconsin-Madison	100	U. of Wisconsin-Madison	195	U. of Illinois at U-C	320
	Colombia U.	38	Colombia U.	100	U. of Michigan Ann Arbor	193	U. of Michigan Ann Arbor	319
	Cornell U.	36	U. of Michigan Ann Arbor	96	U. of California-Berkeley	191	Arizona State U.	308
	Carnegie-Mellon U.	34	U. of Arizona	96	Arizona State U.	187	Cornell U.	306
	U. of Wisconsin-Madison	34	MIT	92	Purdue U.	187	U. of California-Berkeley	301
Weighted node degree	U. of California-San Diego	90	U. of Minnesota-Twin C.	256	U. of Washington	706	Colombia U.	1,197
	U. of Washington	73	U. of Washington	213	U. of Colorado at Boulder	606	U. of Illinois at U-C	1,183
	U. of Illinois at U-C	65	U. of Illinois at U-C	197	U. of Illinois at U-C	604	U. of Washington	1,152
	MIT	63	MIT	185	Pennsylvania State U.	599	MIT	1,136
	U. of Michigan Ann Arbor	60	Colombia U.	182	Colombia U.	562	U. of Colorado at Boulder	1,120
	U. of Wisconsin-Madison	60	U. of California-Berkeley	173	U. of Wisconsin-Madison	558	U. of Michigan Ann Arbor	1,050
	Colombia U.	55	U. of Michigan Ann Arbor	169	U. of California-Berkeley	549	Pennsylvania State U.	1,040
	Cornell U.	55	U. of Arizona	163	U. of Michigan Ann Arbor	537	Cornell U.	1,035
	U. of California-Berkeley	55	U. of Colorado at Boulder	162	MIT	529	U. of California-Berkeley	1,107
	U. of Georgia	50	U. of Wisconsin-Madison	155	U. of Arizona	520	U. of Wisconsin-Madison	992

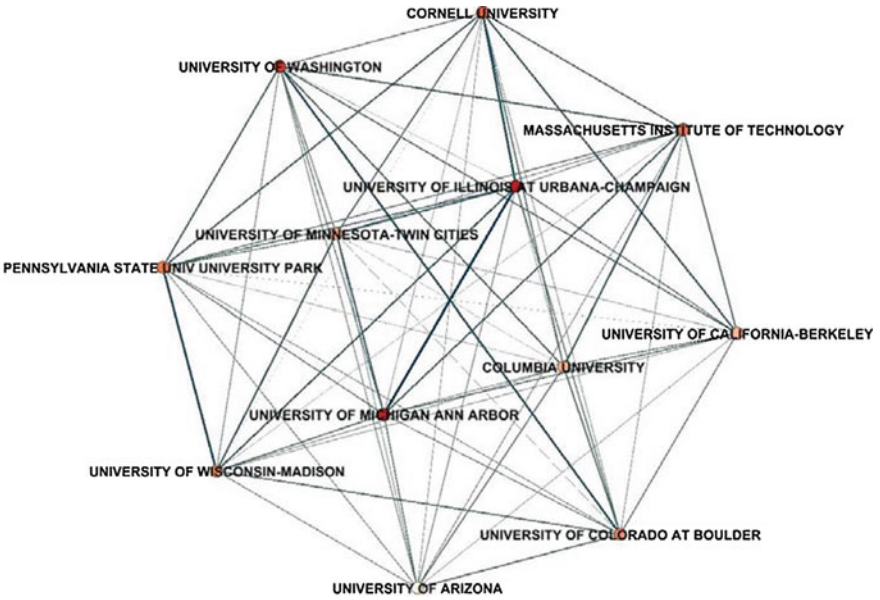


Fig. 7 High node degree organizations' collaboration network

In order to illustrate the collaboration of organizations with the highest number of collaborative grants, we draw the network of the top 10 organizations in Fig. 7. This network forms a clique, i.e., all organizations collaborated in grants with the others. The thickness of the edges presents the number of collaborations among these organizations. The boldness of the color of each node represents the weighted node degree for that node. The highest number of collaborations is between the University of Washington and the Arizona State University with 27 grants. The lowest collaboration among this group is between the Arizona State University and the Columbia University with 5 grants.

Moreover, in order to study frequent collaborations, we only consider edges where there are more than 10 collaborations in Fig. 8. As seen in the figure, the ratio of the distinct pairs is lower than that of the PIs' frequent collaboration network in Fig. 5. There are more triangles and even larger cliques in this network indicating frequent collaboration among those organizations.

4.2.1 Historical Perspective

Above, we analyze organization collaboration network of the NSF grants from 1976 to 2011. However, in order to capture the changes within this network and to analyze the evolution of the network better, one needs to analyze the network at different time frames. Therefore, in this section, we generate the organization collaboration

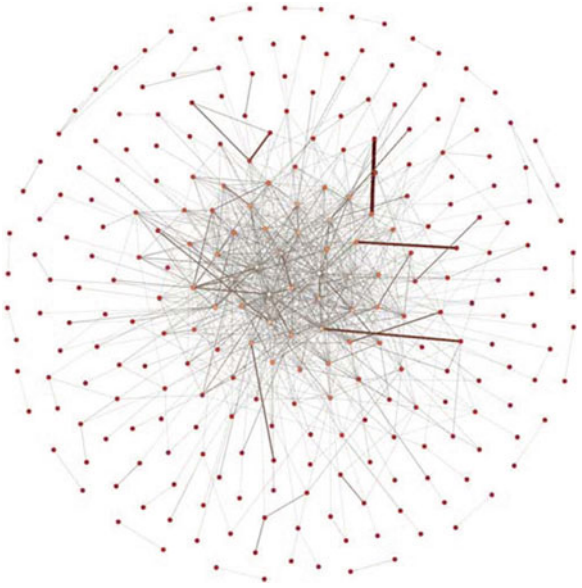


Fig. 8 Organization frequent collaboration network

Table 2 Organization network characteristics over years

	80s	90s	00s	Overall
Avg. degree	5.39	7.36	15.3	15.85
Avg. W. degree	7.50	11.03	27.9	33.36
Diameter	9	9	7	6
Avg. path length	3.54	3.53	3.1	3.07
Avg. clustering coef.	0.15	0.19	0.32	0.34

network for 1980s, 1990s and 2000s. For 2000s, we just analyze the grants awarded from 2000 to 2009 in order to have the same time frame length in each network.

Table 2 represents the characteristics of organization collaboration networks of 1980s, 1990s and 2000s. According to this table, there is a steep increase in the average node degree and the average weighted node degree. The average node degrees of the networks are 5.39, 7.36 and 15.3, respectively, while the average weighted degrees of the networks are 7.5, 11.03, and 27.9, respectively. These values clearly illustrate that both the average number of collaborators and the total number of collaborations with other peer institutions have increased considerably. Additionally, the average number of collaborations made by an organization with its peers has become 1.8, while it was 1.4 in 1980s.

Parallel to the increase in the node degree, the organization network has become denser over the years. The diameter of the network is 9, 9, and 7, respectively for 1980s, 1990s, and 2000s. However, when we look at the overall network of the

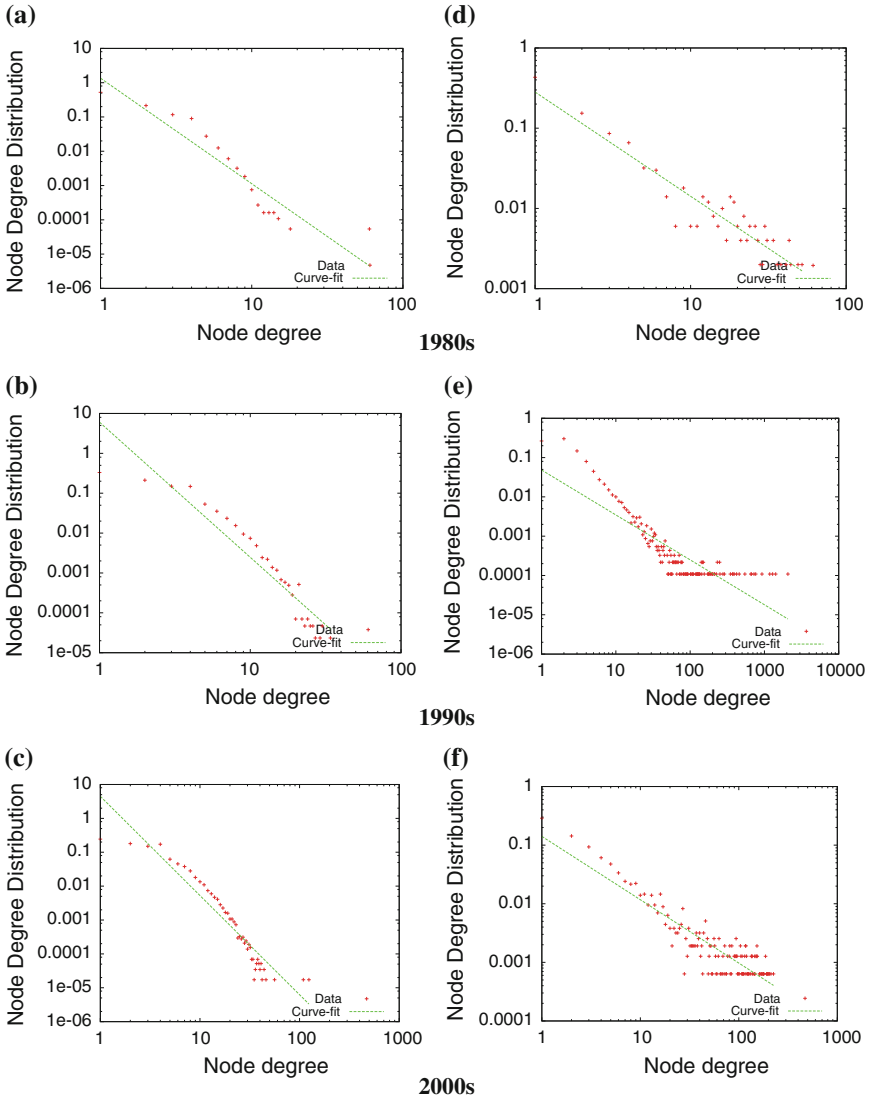


Fig. 9 PI and organization degree distributions (historic). **a** PI degree distribution. **b** PI degree distribution. **c** PI degree distribution. **d** Organization degree distribution. **e** Organization degree distribution. **f** Organization degree distribution

organization collaborations, the diameter is 6. Thus, the *six degrees of separation* has persisted in the organization collaboration network though the past three decades. Moreover, the average path length of the network decreases over the years, while the average clustering coefficient rises. In addition to the *small-world* characteristic of the organization collaboration network, it has become denser over the years as observed in typical social networks.

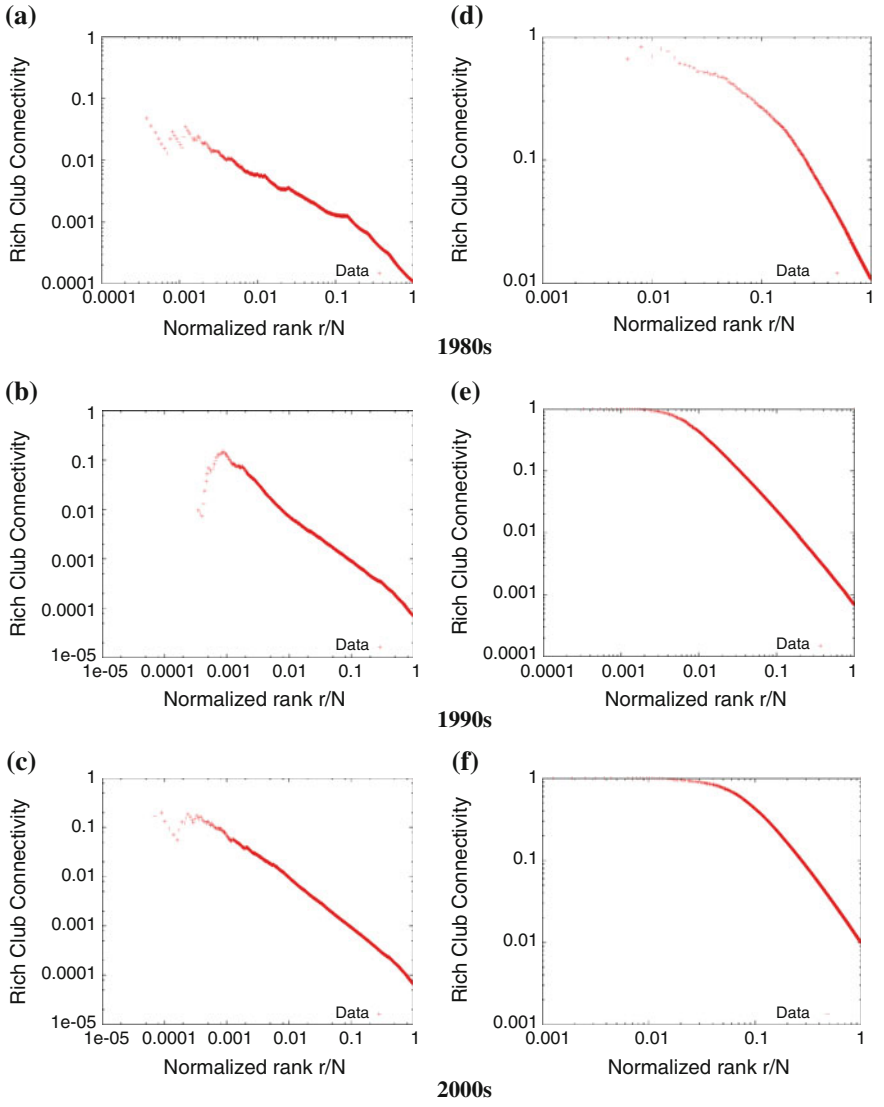


Fig. 10 PI and organization network rich club metric (historic). **a** PI rich club. **b** PI rich club. **c** PI rich club. **d** Organization rich club. **e** Organization rich club. **f** Organization rich club

Table 1 shows the rankings of the top 10 organizations based on the betweenness centrality and node degree values for the 1980s, 1990s, 2000s, and overall (i.e., including all three periods in the network formation) networks. These top 10 organizations are part of the rich club in the network. According to the table, we can conclude that the *rich-get-richer* phenomenon is observed in the organization collaborations networks. Finally, Figs. 9 and 10 present several network characteristics of the Organization and PI collaboration networks for different time frames.

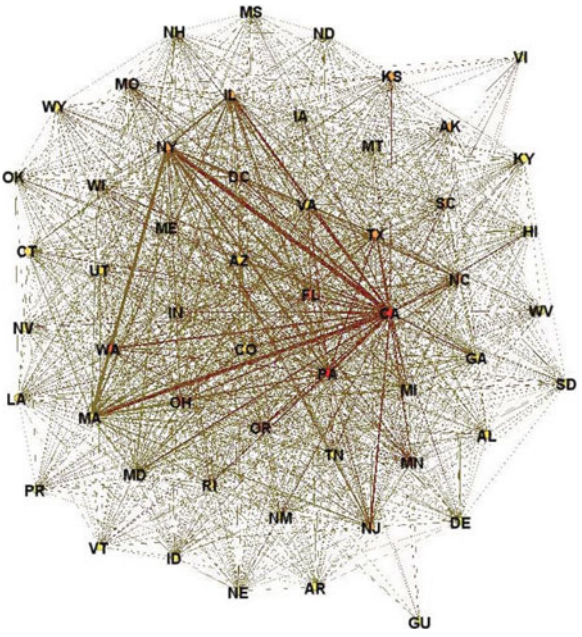


Fig. 11 State collaboration network based on betweenness centrality

4.3 State Network

In order to analyze the patterns among the home state of researchers, we construct the *state network*, i.e., the collaboration network between the states of the PIs. Figure 11 illustrates the state network constructed from the dataset where the nodes with higher betweenness centrality are located towards the center. In this network, there are 54 nodes and 1,285 edges. This network is highly clustered as the maximal clique size is 35 indicating that 35 states pairwise collaborate with each other. The assortativity coefficient is -0.13 for this network. The diameter of the network is 2 and average path length is 1.1. The average node degree of the network is 47.6 and the clustering coefficient is 0.95. All these metrics indicate a highly connected network.

There is no rich club in this network as almost all nodes are well connected. However, we can see the states that have many connections with higher degrees and weights represented with thick lines in the network. For instance, there is a frequent collaboration triangle between the states of New York (NY), California (CA) and Massachusetts (MA), which points to a large number of collaboration among these three states.

Furthermore, we tabulate the betweenness centrality, and weighted node degree for each node in Table 3. According to the table, betweenness centrality values are very close to each other for the top 5 collaborative states. However, average weighted node degree results indicate some differences among the top collaborative states.

Table 3 Top 10 states

Metric	State rankings							
	1980s		1990s		2000s		Overall	
	State	Value	State	Value	State	Value	State	Value
Weighted node degree	CA	379	CA	1,270	CA	3,982	CA	7,730
	NY	284	NY	1,024	NY	2,716	NY	5,946
	PA	187	MA	814	MA	2,384	MA	4,748
	MA	179	PA	605	PA	1,764	PA	3,774
	IL	166	IL	512	IL	1,594	IL	3,289
	TX	137	TX	465	TX	1,563	TX	3,178
	WA	101	MI	402	CO	1,330	CO	2,520
	MI	94	MD	399	FL	1,166	MI	2,333
	FL	93	NC	363	VA	1,105	FL	2,309
	CO	90	CO	333	MI	1,099	NC	2,146
Betweenness centrality	CA	204.7	PA	75.8	FL	21.8	CA	8.2
	VA	77.2	HI	54.6	OR	20.6	NC	8.2
	NY	69.9	CA	26.1	WA	20.5	NY	8.2
	IL	59.3	NY	26.1	CA	12.8	OH	8.2
	FL	48.8	MA	23.2	IL	12.8	TX	8.2
	MA	36.6	NC	20.2	NC	12.8	FL	4.9
	PA	34.3	CO	19.5	NY	12.8	TN	4.9
	CO	34.1	TX	19.2	SC	9.6	MI	4.7
	NC	33.5	MI	16.3	AK	8.9	PA	4.6
	TX	32.1	WA	15.3	KS	7.1	IL	4.6

California (CA) is the most collaborative state with 7,730 inter-state collaborations. Since the node degrees are very close to each other we don't tabulate them. California (CA), North Carolina (NC), Ohio (OH), Pennsylvania (PA) and Texas (TX) have a node degree value of 53; which indicates that they have collaborated with all other states in at least one grant. On the other hand, Virgin Islands (VI), Guam (GU), Puerto Rico (PR), Wyoming (WY), South Dakota (SD), and Mississippi (MS) has collaborated with 13, 14, 35, 40, 41, 42, and 43 states, respectively, and are the states with the smallest node degrees.

Moreover, we analyze frequent collaborations among the states. In Fig. 12, we draw the state collaboration network when the number of collaborations is greater than 250. There are 10 states which collaborated in more than 250 grants. As seen in the figure, California (CA) collaborated at least 250 times with all the other states in this network. The high collaboration among NY, CA and MA is more visible in this figure.

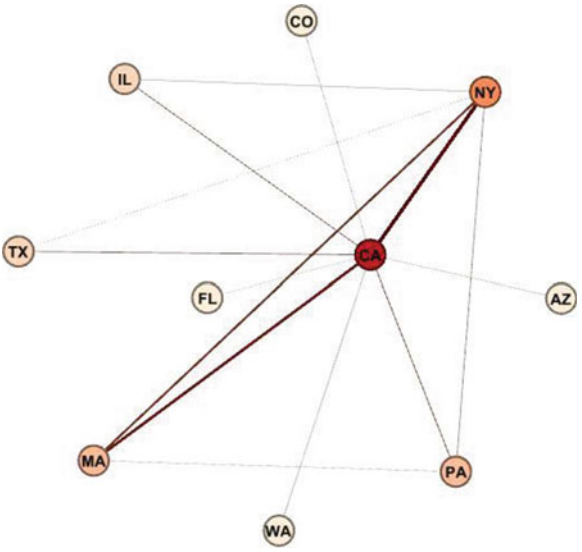


Fig. 12 State frequent collaboration network

4.3.1 Historical Perspective

Table 4 represents the network characteristics of state collaboration networks of 1980s, 1990s and 2000s, respectively. According to this table, there is a considerable increase in the average node degree and the average weighted node degree values. The average node degrees of the networks are 17.9, 34.2 and 43.3, respectively while the average weighted degrees of the networks are 57.5, 229.6, and 695.1, respectively. These values clearly illustrate that inter-state research collaboration has increased over the years. Additionally, the average number of collaborations made by a state with its peers has become 16 in 2000s, while it was around 3 in 1980s. Thanks to this increase in the node degree, the overall state collaboration network has become almost a *clique*, i.e., full mesh. The diameters of the networks are 3, 3, and 3, respectively over the three decades. This is mainly due to two states, namely Virgin Islands (VI) and Guam (GU), which have very low collaborations. They don't have a research collaboration and a common collaborator in given time frames. However, when we look at the overall network of the organization collaborations, the diameter reduces to 2. Moreover, average path length of the network decreases over the years and has become 1.09 while the average clustering coefficient rises and has become 0.95 in the overall network.

Table 3 shows the rankings of the top 10 states based on the weighted node degree and the betweenness centrality values for the 1980s, 1990s, 2000s, and overall networks. The top 10 states have slightly changed over the years. Additionally, according to this table, we can conclude that *rich-get-richer* phenomenon applies to the state collaborations network, as well.

Table 4 State network characteristics over years

	80s	90s	00s	Overall
Avg. degree	17.9	34.15	43.3	47.7
Avg. W. degree	57.5	229.6	695.1	1405.0
Diameter	3	3	3	2
Avg. path length	1.69	1.37	1.18	1.09
Avg. clustering coef.	0.63	0.78	0.81	0.95

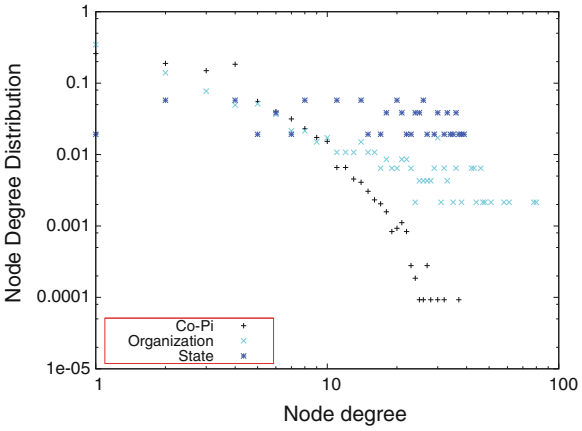


Fig. 13 Degree distribution for CSE directorate

4.4 Directorates Networks

In the previous sections, we construct three kinds of networks based on the whole NSF funding data. In this section, we construct these networks for each directorate separately to analyze the funding structures within each NSF directorate. The dataset contains 9 different NSF directorates, namely: Biological Sciences (BIO), Computer and Information Sciences (CSE), Education and Human Resources (EHR), Engineering (ENG), Geosciences (GEO), Mathematical and Physical Sciences (MPS), Office of Polar Programs (OPP), Social Behavioral and Economic Sciences (SBE), and Office of the Director (OD).

By considering each directorate we calculate node degree values of the PI, the organization, and the state networks. The graphs for node degree distributions of each directorate are shown in Figs. 13 and 14. When considering each directorate individually, the corresponding networks do not have a rich club similar to the whole network. Additionally, the assortativity value of each individual directorate network is close to zero indicating indifference to the popularity of the peers.

According to the clustering coefficient values of the directorate networks, GEO directorate has the highest clustering coefficient in the state network followed by BIO and ENG. These three directorates have the highest clustering coefficient values in the PI and the organization networks, as well, which indicates that the collaboration

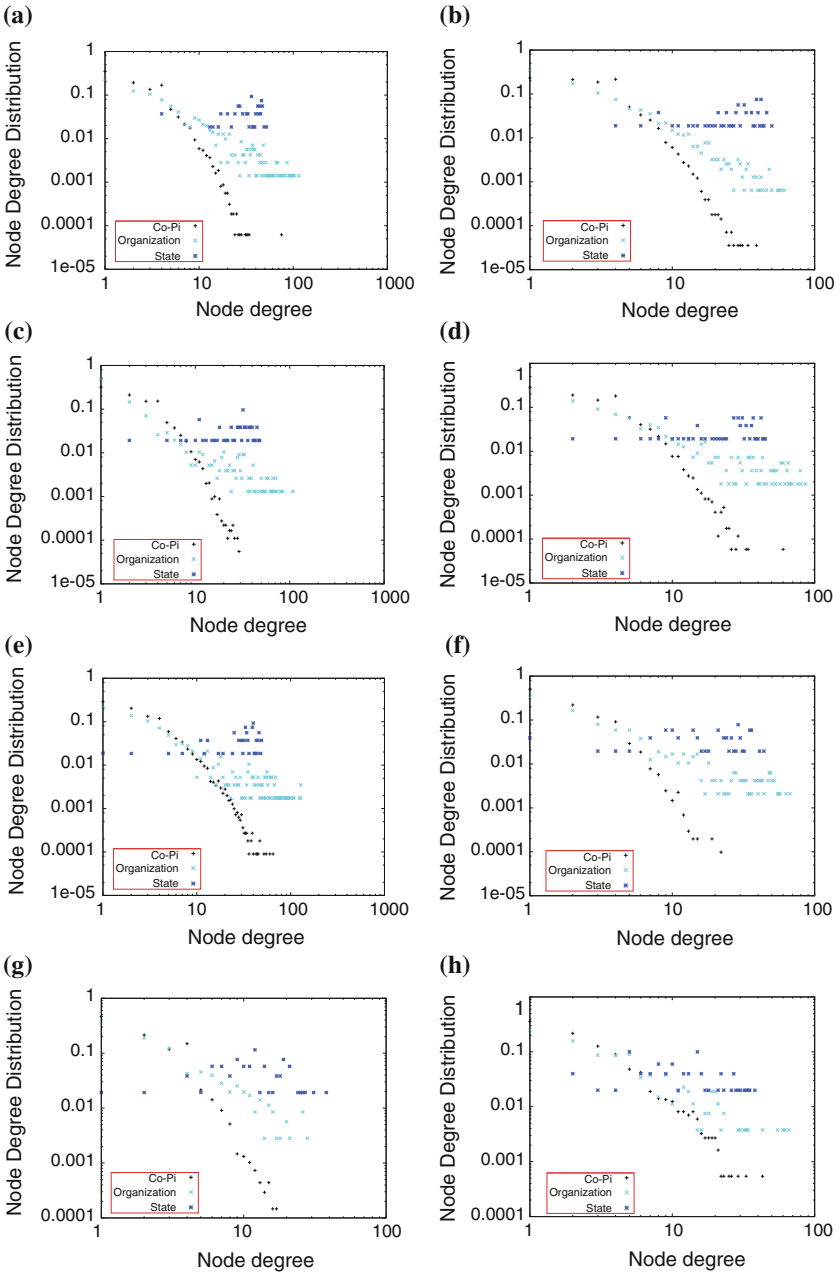


Fig. 14 Metrics for directorates networks. **a** Degree distribution for BIO directorate. **b** Degree distribution for EHR directorate. **c** Degree distribution for EHR directorate. **d** Degree distribution for MPS directorate. **e** Degree distribution for GEO directorate. **f** Degree distribution for SBE directorate. **g** Degree distribution for OD directorate. **h** Degree distribution for OPP directorate

Table 5 Network characteristics for different project sizes

	Small P.	Medium P.	Large P.	Overall
Avg. degree	9.46	6.42	14.6	15.9
Avg. W. degree	13.3	16.2	21.1	33.3
Diameter	6	8	6	6
Avg. path length	2.97	2.88	2.60	3.07
Avg. clustering coef.	0.36	0.50	0.59	0.34

within these directorates are much more emphasized than the other NSF directorates. It also indicated, however, that researchers whose home directorate is one of these three directorates have a lower likelihood of collaborating with researchers from other directorates.

Additionally, as expected, the PI networks of directorates are better clustered than the overall PI network. Their diameter and average shortest path values are much smaller than those of the overall PI network, as well.

4.5 Project Size

NSF categorizes the research projects based on funding levels. There are mainly three types of projects: small projects (typically, <500K), medium projects (typically, 500K-2M), and large projects (typically, >2M). In order to analyze the collaboration patterns within different project sizes, we generate the organization networks to investigate the collaboration among organizations at different funding levels (Fig. 15).

Table 5 represents the network characteristics of organization collaboration networks of small, medium and large projects. According to the table, the average node degrees of the networks are 9.46, 6.42 and 14.6, respectively. Interestingly, organizations collaborated with more different peers in smaller projects than the medium projects. The average weighted degrees of the networks are 13.3, 16.2, and 21.1, respectively. Accordingly, the average number of collaborations made by an organization with its peers is 1.4 in small and large projects while it is 2.5 in medium projects. This also indicates that organizations collaborate with more peers in small and large projects. However, the average number of collaborations made by an organization with its peers is higher in medium projects, indicating more persistent collaborations at medium level of funding.

The diameters of the networks are 6, 8, and 6, respectively. Since the average number of collaborators of an organization is the lowest in medium project network, this network has the highest diameter. Moreover, in the large project collaboration network, we have the lowest average path length and the highest average clustering coefficient values. Thus, while all networks are *small-worlds*, the large project collaboration network exhibits the *small-world* characteristics more than the other funding levels.

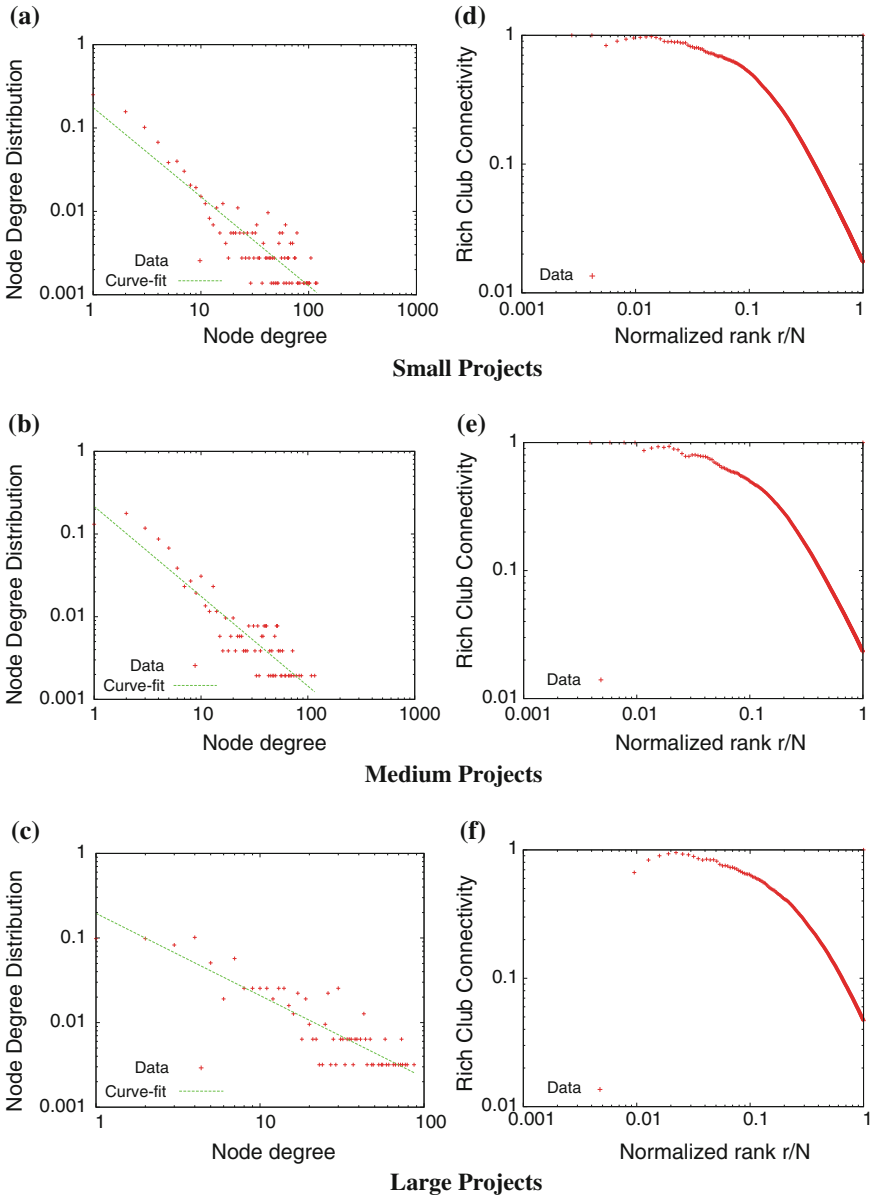


Fig. 15 Organization network characteristics of different project sizes. **a** Degree distribution. **b** Degree distribution. **c** Degree distribution. **d** Rich club. **e** Rich club. **f** Rich club

5 Conclusion and Future Work

In this chapter, we analyzed publicly available data on NSF funded grants to reveal the collaboration patterns among researchers. We derived three different kinds of networks to analyze the trends within the funding of the PI network, the organization network, and the state network. The PI network reveals a small-world characteristic but does not exhibit a power-law degree distribution. However, organization network exhibits a power-law degree distribution with a rich club that has majority of the collaborations. The state network is highly clustered but we identified the most central states in terms of collaborations. We construct these networks both for different time frames and as a whole in order to compare the network characteristics of these networks for different time frames and capture the evolution of the NSF collaboration network over the time. We further analyze the funding network within each NSF directorate and find that some research fields are more collaborative than others in obtaining federal funding. Finally, we analyze these networks for small, medium, and large project sizes in order to observe the collaboration at different funding levels.

Our study revealed several interesting findings while reaffirming some of the anticipated characteristics of the funding network. We clearly observed a six degrees of separation in the state and the organization collaboration networks, while the degree of separation in the PI network is much higher. Another observation was that most of the funded collaborative projects had only two PIs.

Several extensions to the grant network analysis is of interest. In our study, we focussed on the successful grant proposals. To obtain a better picture of the collaboration patterns in the research funding, it would be very helpful to consider unsuccessful proposals. Furthermore, NSF uses different recommendation levels to rank grant proposals, e.g., Highly Recommended, Recommended, or Low Recommended. Consideration of these recommendation levels while constructing the collaboration networks would reveal more refined patterns. However, the challenge is to obtain such data without violating the privacy of PIs. Lastly, it would be interesting to observe the collaboration patterns in agencies other than the NSF and the United States.

References

1. Newman M, Barabasi A-L, Watts DJ (2006) The structure and dynamics of networks: (Princeton studies in complexity). Princeton University Press, Princeton
2. Gjoka M, Kurant M, Butts CT, Markopoulou A (2010) Walking in Facebook: a case study of unbiased sampling of OSNs. In: Proceedings of IEEE INFOCOM '10, San Diego, CA, March 2010
3. Cha M, Kwak H, Rodriguez P, Ahn YY, Moon S (2007) I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In: Proceedings of the 7th ACM SIGCOMM conference on Internet measurements, ser. IMC '07. ACM, New York, pp 1–14
4. Ediger D, Jiang K, Riedy EJ, Bader DA, Corley C, Farber R, Reynolds WN (2010) Massive social network analysis: mining twitter for social good. In: 39th international conference on parallel processing (ICPP), San Diego, CA, Sept 2010

5. Java A, Song X, Finin T, Tseng B (2007) Why we twitter: understanding microblogging usage and communities. In: WebKDD/SNA-KDD '07: Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on web mining and social network analysis. ACM, New York, pp. 56–65
6. Newman MEJ (2001) Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality. Phys Rev E 64(1):016132
7. Igraph, http://igraph.sourceforge.net/
8. Arslan E, Gunes MH, Yuksel M (2011) Analysis of academic ties: a case study of mathematics genealogy. In: IEEE Globecom workshop on complex and communication networks, Houston, TX, 9 Dec 2011
9. Cheung D, Gunes MH (2012) A complex network analysis of the United States air transportation. In: IEEE/ACM international conference on Advances in social networks analysis and mining (ASONAM), Istanbul, Turkey, 26–29 Aug 2012
10. Cotta C, Merelo J-J (2006) The complex network of EC authors. ACM SIGEVOlution 1:2–9
11. Nerur S, Sikora R, Mangalaraj G, Balijepally V (2005) Assessing the relative influence of journals in a citation network. Commun ACM 48:71–74
12. Zhang P, Koppaka L (2007) Semantics-based legal citation network. In: ACM proceedings of the 11th international conference on artificial intelligence and law, Palo Alto, CA, 4–8 June 2007, pp 123–130
13. Naruchitparames J, Gunes MH, Louis S (2011) Friend recommendations in social networks using genetic algorithms and network topology. In: IEEE congress on evolutionary computation, New Orleans, LA, June 2011, pp 5–8
14. Breland AE, Gunes MH, Schlauch KA, Harris FC (2010) Mixing patterns in a global influenza a virus network using whole genome comparisons. In: IEEE computational intelligence in bioinformatics and computational biology (CIBCB), Montreal, Canada, May 2010, pp 2–5
15. Breland AE, Schlauch KA, Gunes MH, Harris FC (2010) Fast graph approaches to measure influenza transmission across geographically distributed host types. In: ACM international workshop on graph theoretic approaches for biological network analysis (IWBNA 2010), Niagara Falls, NY, 2–4 Aug 2010
16. Akgun MB, Gunes MH (2011) Link-level network topology generation. In: ICDCS workshop on simplifying complex networks for practitioners, Minneapolis, MN, 24 June 2011
17. Kardes H, Gunes M, Oz T (2012) Cheleby: a subnet-level internet topology mapping system. In: Fourth international conference on communication systems and networks (COMSNETS), Jan 2012, pp 1–10
18. Akgun MB, Gunes MH (2013) Bipartite internet topology at the subnet-level. In: IEEE international workshop on network science (NSW 2013), West Point, NY, 29 Apr–1 May 2013
19. Ryfe DM, Mensing D, Ceker H, Gunes MH (2012) Popularity is not the same thing as influence: a study of the bay area news system. In: Journal of the international symposium of online journalism (#ISOJ), vol 2(2), Fall 2012
20. Ramos D, Gunes MH, Mensing D, Ryfe DM (2013) Mapping emerging news networks: a case study of the San Francisco bay area. In: Studies in computational intelligence: complex networks, vol 424, pp 237–244
21. Li X, Chen H, Zhang Z, Li J (2007) Automatic patent classification using citation network information: an experimental study in nanotechnology. In: Proceedings of the 7th ACM/IEEE-CS joint conference on digital libraries, Vancouver, BC, Canada, 18–23 June 2007, pp 419–427
22. Hall BH, Jaffe AB, Trajtenberg M (2001) The nber patent citations data file: lessons, insights and methodological tools. In: NBER working papers 8498, National Bureau of Economic Research Inc
23. Komurov K, Gunes MH, White MA (2009) Fine-scale dissection of functional protein network organization by statistical network analysis. PLoS ONE 4(6)
24. Dittrich A, Gunes MH, Dascalu S (2013) Network analysis of software repositories: identifying subject matter experts. In: Studies in computational intelligence: complex networks, vol 424, pp 187–198

25. Zachor C, Gunes MH (2013) Software collaboration networks. In: Studies in computational intelligence: complex networks, vol 424, pp 257–264
26. Morelli T, Gunes MH (2012) Video game industry as a complex network. In: 2nd workshop on social network analysis and applications (SNAA 2012), Istanbul, Turkey, 26 Aug 2012
27. Kardes H, Sevincer A, Gunes MH, Yuksel M (2012) Six degrees of separation among US researchers. In: IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM 2012), Istanbul, Turkey, 26–29 Aug 2012
28. National Science Foundation, http://www.nsf.gov/about/
29. National Science Foundation Award Search, http://www.nsf.gov/awardsearch/
30. Herr II, Bruce W, Talley EM, Burns GAPC, Newman DLR (2009) Gavin, "Interactive science map of nih funding". http://scimaps.org/maps/nih/2007/
31. Chun B, Culler D, Roscoe T, Bavier A, Peterson L, Wawrzoniak M, Bowman M (2003) Planetlab: an overlay testbed for broad-coverage services. SIGCOMM Comput Commun Rev 33:3–12
32. Bastian M, Heymann S, Jacomy M (2009) Gephi: an open source software for exploring and manipulating networks
33. Watts DJ, Strogatz SH (1998) Collective dynamics of 'small-world' networks. Nature 393(6684):440–442
34. Newman MEJ (2002) Assortative mixing in networks. Phys Rev Lett 89(20):208701+
35. McAuley JJ, da Fontoura Costa L, Caetano TS (2007) Rich-club phenomenon across complex network hierarchies. Appl Phys Lett 91:084103

A Density-Based Approach to Detect Community Evolutionary Events in Online Social Networks

Muhammad Abulaish and Sajid Yousuf Bhat

Abstract With the advent of Web 2.0/3.0 supported social media, Online Social Networks (OSNs) have emerged as one of the popular communication tools to interact with similar interest groups around the globe. Due to increasing popularity of OSNs and exponential growth in the number of their users, a significant amount of research efforts has been diverted towards analyzing user-generated data available on these networks, and as a result various community mining techniques have been proposed by different research groups. But, most of the existing techniques consider the number of OSN users as a fixed set, which is not always true in a real scenario, rather the OSNs are dynamic in the sense that many users join/leave the network on a regular basis. Considering such dynamism, this chapter presents a density-based community mining method, *OCTracker*, for tracking overlapping community evolution in online social networks. The proposed approach adapts a preliminary community structure towards dynamic changes in social networks using a novel density-based approach for detecting overlapping community structures and automatically detects evolutionary events including *birth*, *growth*, *contraction*, *merge*, *split*, and *death* of communities with time. Unlike other density-based community detection methods, the proposed method does not require the neighborhood threshold parameter to be set by the users, rather it automatically determines the same for each node locally. Evaluation results on various datasets reveal that the proposed method is computationally efficient and naturally scales to large social networks.

M. Abulaish (✉) · S. Y. Bhat
Department of Computer Science, Jamia Millia Islamia, New Delhi 110025, India
e-mail: abulaish@ieee.org

S. Y. Bhat
e-mail: s.yousuf.jmi@gmail.com

F. Can et al. (eds.), *State of the Art Applications of Social Network Analysis*,
Lecture Notes in Social Networks, DOI: 10.1007/978-3-319-05912-9_9,

1 Introduction

With increasing popularity of Online Social Networks (OSNs) and their wide applications in different walk of life, community mining research has attracted researchers from various fields including data mining, web mining, and network science in recent past and the field is still rapidly evolving. As a result, various methods based on spectral clustering [1, 2], partitional clustering [3], modularity optimization [4], and latent space clustering [5] have been developed to identify users' communities in social networks. The fact that a person may have different diverse interests and consequently she may participate in more than one community has resulted in an increased attention towards detecting overlapping communities in social networks, and a solution based on k-clique percolation given by Palla et al. [6] is a step towards this end, followed by other density-based community detection methods, including `gSkeletonClu` [7], `CHRONICLE` [8], and `CMiner` [9] that are based on `DBSCAN` [10].

One of the important properties of the real-world social networks is that they tend to change dynamically as most often: (1) new users join the network, (2) old users leave the network, and (3) users establish/break ties with other users. Consequently, all these evolutionary events result in *birth*, *growth*, *contraction*, *merge*, *split*, and *death* of communities with time. Although a number of community finding techniques have been proposed by different researchers, the dynamic nature of the real-world social networks (specifically, the online social networks like Facebook and Twitter) has been largely ignored in terms of community detection. In case of dynamic social networks, most of the studies either analyze a single snapshot of the network or an aggregation of all interactions over a possibly large time-window. But, such approaches may miss important tendencies of dynamic networks and in fact the possible causes of this dynamic behavior may be among the most important properties to observe [11]. Although, recent literature includes some approaches for analyzing communities and their temporal evolution in dynamic networks, a common weakness in these studies is that communities and their evolutions have been studied separately. As pointed out in [12], a more appropriate approach would be to analyze communities and their evolution in a unified framework, where community structure provides evidence about community evolution.

Considering the case of OSNs like Facebook and Twitter, community structures have mostly been analyzed using traditional community detection techniques over social networks representing explicit relations (friends, colleagues, etc.) of users. However, the observations made by Wilson et al. [13] and Chun et al. [14] on Facebook friendship and interaction data reveals that for most users, majority of their interactions occur only across a small subset of their social links, proving that only a subset of social links actually represents interactive relationships. Their findings suggest that social network-based systems should be based on the activity network, rather than on the social link network.

This chapter presents the design of a density-based unified method, *OCTracker*, to identify overlapping communities and track their evolution in online social networks.

The initial version of this work has been published as a short chapter in proceedings of the ASONAM'12 [15], and the major enhancement is the enhancement of the proposed methodology and the addition of more experimental results on different datasets. The proposed method detects dynamic overlapping community structures by automatically highlighting evolutionary events like *birth*, *growth*, *contraction*, *merge*, *split*, and *death* with time using a density-based approach. The novelty of the method lies in its overlapping community detection approach, which does not require the neighborhood threshold ε (mostly difficult to determine for density-based community detection methods) to be specified by the users manually. In addition, the proposed method is scalable to large social networks.

The rest of the chapter is organized as follows. Section 2 presents a brief review of the related works. Section 3 defines the similarity function and presents the density-based overlapping community detection approach. Section 4 describes the proposed approach for tracking evolution of overlapping communities in dynamic social networks. Section 6 presents the parameter estimation process, followed by a brief explanation of the overlapping community merging process in Sect. 5. Section 7 presents the experimental setup and evaluation results. Finally, Sect. 8 concludes the chapter.

2 Related Work

Traditional community finding approaches are generally based on either graph partitioning methods [16] or partition-based clustering [17, 18], where the problem is to divide the nodes into k clusters by optimizing a given cost function. However, the main drawback of these methods lie in the requirement of the number of clusters and their sizes a priori. Hierarchical clustering is another well-known technique used in social network analysis [19, 20]. Starting from a partition in which each node is in its own community or all nodes are in the same community, one merges or splits clusters according to a topological measure of similarity between nodes. Other similar methods include methods based on the sociological notion of betweenness centrality [21] and methods based on modularity Q optimization [4].

Extending the `DBSCAN` (Density-Based Spatial Clustering of Applications with Noise) algorithm [10] to undirected and un-weighted graph structures, Xu et al. [22] proposed `SCAN` (Structural Clustering Algorithm for Networks) to find clusters, hubs, and outliers in large networks based on structural similarity, which uses the neighborhood of vertices as clustering criteria. `CHRONICLE` [8] is a two-stage extension of `SCAN` to detect the dynamic behavior of communities in dynamic networks. Similarly, considering only the weighted interaction graph of the online social networks, Falkowski et al. [23] extended the `DBSCAN` algorithm [10] to weighted interaction graph structures of online social networks. Some important features of density-based community detection methods include less computation, detection of outliers and natural scalability to large networks. However, the main drawback of traditional density-based community detection methods is that they require the global

neighborhood threshold, ε, and the minimum cluster size, μ, to be specified by the users. The methods are particularly sensitive to the parameter, (ε), which is difficult to determine. As an alternative, the method proposed in [7] reduces the number of possible values to consider for ε significantly by considering only the edge weights of a Core-Connected Maximal Spanning Tree (CCMST) of the underlying network.

The most popular method for identifying overlapping communities is the Clique Percolation Method (CPM) proposed by Palla et al. [6], which is based on the concept of k-clique, i.e., a complete subgraph of k nodes. The method relies on the observation that communities seem to consist of several small cliques that share many of their nodes with other cliques in the same community. In [24], the authors presented an overlapping community detection method `MOSES` by combining local optimization with Overlapping Stochastic Block Modeling (OSBM) [25] using a greedy maximization strategy. Here communities are created and deleted, and nodes are added or removed from communities, in a manner that maximizes a likelihood objective function.

In order to find communities in dynamic social networks and to track their evolutions, various methods have been proposed recently. A typical dynamic community detection problem is formulated in [11, 26]. In these works, along a discrete timescale and at each time-step, social interactions of certain individuals of a network are observed and several subgraphs are formed. Based on these subgraphs, the true underlying communities and their developments over time are identified, so that most of the observed interactions can be explained by the inferred community structure. Similar approaches have been followed in [8, 27, 28]. However, as pointed out in [12], a common weakness in these approaches is that communities and their evolution are studied separately. It would be more appropriate to analyze communities and their evolution in a unified framework where community structure provides evidence about the community evolutions. Along this direction, [29] proposed a framework for studying community dynamics where a preliminary community structure adapts to dynamic changes in a social network. Our approach is similar to [29], but unlike it, our concern is on tracking the evolution of overlapping communities and we do not need an ageing function to remove old interactions from the network. Moreover, our method is applicable to directed/un-directed and weighted/un-weighted networks, whereas [29] applies only to un-directed and weighted networks. For un-weighted networks, the proposed method considers a unit weight for each edge in the network without altering the meaning or representation of the network.

3 Proposed Method

In this section we present the procedural detail of the proposed method to identify community evolution events. Along the lines of the `SCAN` [22], `DENGRAPH` [23], and other density-based community detection methods like `gSkeletonClu` [7] and `CHRONICLE` [8], the proposed method is based on `DBSCAN` [10]. As pointed out in Sect. 2, the main drawback of traditional density-based community detection methods

Table 1 Notations and their descriptions

Notation	Description
V	Set of nodes in the social network
E	Set of links in the social network
$I_{\overrightarrow{p}}$	Total number of out-going interactions of a node p
$I_{\overrightarrow{pq}}$	Number of interactions from node p to node q
$I_{\overleftrightarrow{pq}}$	Reciprocated interactions of p and q: $\min(I_{\overrightarrow{pq}}, I_{\overrightarrow{qp}})$
$I_{\overleftrightarrow{p}}$	Number of reciprocated interactions of a node p: $\sum_{\forall q \in V_p} \min(I_{\overrightarrow{pq}}, I_{\overrightarrow{qp}})$
V_p	Set of nodes in the network with whom node p interacts
V_{pq}	Set of nodes with whom both nodes p and q interact: $V_p \cap V_q$

is that they require the global neighborhood threshold, ε, and the minimum cluster size, μ, to be specified by the users. On the other hand, the proposed method does not require the global neighborhood threshold parameter, ε, to be set manually at the beginning of the process. Instead, it uses a local representation of the neighborhood threshold which is automatically calculated for each node locally using a much simpler approach from the underlying social network. Moreover, a local version of μ is also computed for each node automatically using a global percentage parameter η. The proposed method thus requires only a single tunable parameter η to be set by the users.

3.1 Distance Function and Parameter Estimation

This section presents a formal definition of a novel distance function and related concepts that are used in the proposed density-based overlapping community finding algorithm. The distance function defines distance between a pair of nodes by taking into consideration the average number of reciprocated interactions between the nodes and their commonly interacted nodes in the network. Considering the social network as a graph $G = (V, E)$, where V is the set of nodes representing users and $E \subseteq V \times V$ is the set of links between the users based on their interactions in the network, the distance function can be defined formally in the following paragraph. For simplicity, the symbols used throughout this chapter and their interpretations are presented in Table 1.

Definition 1 (*Distance*) For any two interacting nodes $p, q \in V$, the distance between them is represented as $\Delta(p, q)$ and defined as the minimum of the reciprocals of their mutual directed responses, normalized by their respective total count of outgoing interactions in the interaction graph, as shown in Eq. 1.

$$\Delta(p,q) = \begin{cases} \min\left(\frac{\delta(p,q)^{-1}}{I_{\overrightarrow{p}}}, \frac{\delta(q,p)^{-1}}{I_{\overrightarrow{q}}}\right) & \text{if } \delta(p,q) > 0 \wedge \delta(q,p) > 0 \\ 1 & \text{otherwise} \end{cases} \tag{1}$$

In Eq. 1, $\delta(p, q)$ represents the response of node q to the interactions of node p, and defined as the average of the per-user reciprocated interactions (link weights) of q and the nodes of V_{pq}, with p, if $I_{\overleftrightarrow{pq}} > 0$, otherwise 0. Mathematically, it can be defined using Eq. 2, where V_{pq} and $I_{\overleftrightarrow{pq}}$ have same interpretations as given in Table 1.

$$\delta(p,q) = \begin{cases} \left(\frac{\sum_{s \in V_{pq}} (I_{\overleftrightarrow{ps}}) + I_{\overleftrightarrow{pq}}}{|V_{pq}|+1} \right) & \text{if } I_{\overleftrightarrow{pq}} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Smaller values for $\Delta(p, q)$ represent higher response between the nodes p and q and thus represent more closeness between p and q, whereas higher values for $\Delta(p, q)$ translates to higher distance and thereby less closeness between the nodes p and q.

Definition 2 (*Local-Neighborhood Threshold*) For a node $p \in V$, the local neighborhood threshold is represented as ε_p and defined using Eq. 3 as the average per-receiver reciprocated interaction-score of p with all its neighbors (i.e., friends and non-friends with whom it interacts).

$$\varepsilon_p = \begin{cases} \frac{\left(\frac{I_{\overleftrightarrow{p}}}{|V_p|} \right)^{-1}}{I_{\overrightarrow{p}}} & \text{if } I_{\overleftrightarrow{p}} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

In Eq. 3, $\frac{I_{\overleftrightarrow{p}}}{|V_p|}$ represents the average number of reciprocated interactions between a node p and all other nodes in V to whom p sends interactions. The denominator $I_{\overrightarrow{p}}$ represents the total count of outgoing interactions of node p in the interaction graph and it has been used to normalize the value of ε_p to the range [0, 1].

Definition 3 (*Local ε-neighborhood*) The local ε-neighborhood of a node $p \in V$ is represented by $N_{local}\, p$ and defined as the set of nodes to which p sends interactions such that the distance between p and each node in $N_{local}\, p$ is less than or equal to ε_p. Formally, the local ε_p-neighborhood of a node p can be given by Eq. 4.

$$N_{local}\, p = \{q : q \in V_p \wedge distance(p, q) \leq \varepsilon_p\} \tag{4}$$

For our proposed method, we define a local version of minimum-number-of-points for a node p, represented by μ_p, which is also computed automatically from the underlying social network. However, we need to specify a fraction η between [0.0–1.0] to compute μ_p for a node p. For a node $p \in V$, the value of μ_p is taken as the fraction η of its interacted nodes in the network.

It should be noted that the fraction η, forms the only parameter for the proposed method to be set by the users. Moreover, besides determining the local minimum-number-of-points threshold, μ_p, for a node p, the value of η is also used to specify a distance constraint, which is specified as follows. The distance between two

interacting nodes p and q can be measured by Eq. 1 only if the number of commonly interacted nodes of p and q is greater than the number of nodes defined by the fraction η of the minimum of their individually interacted nodes minus one. Otherwise, the distance between them is taken as 1. Formally, the distance constraint can be specified using Eq. 5.

$$distance(p,q) = \begin{cases} \Delta(p,q) & \text{if } |V_{pq}| > (\eta \times \min(|V_p|, |V_q|)) - 1 \\ 1 & \text{otherwise} \end{cases} \tag{5}$$

Definition 4 (*Core node*) A node $p \in V$ having non-zero reciprocated interactions is defined to be a core node with respect to a global percentage constant η, if its local ε_p-neighborhood contains at least μ_p (local minimum-number-of-points threshold for p) of its interacted nodes.

The proposed method identifies core nodes and uses them to grow communities in a recursive manner using the following definitions. It should be noted that all the definitions used in the proposed method are significantly different from the definitions used in traditional density-based community detection methods in terms of the overall concept used to define a community.

Definition 5 (*Direct density-reachability*) A node q is direct density-reachable from a node p with respect to η if p is a core node and q belongs to the local ε_p-neighborhood of p.

Direct density-reachability is an asymmetric relation, i.e., if a node q is direct density-reachable from a node p, then it is not necessarily true otherwise.

Definition 6 (*Mutual cores*) Two nodes p and q are called mutual cores if both p and q are core nodes, and p belongs to the local ε_q-neighborhood of q, and q belongs to the local ε_p-neighborhood of p. In other words, two nodes p and q are mutual cores if they are direct density-reachable from each other.

Definition 7 (*Density reachability*) A node q is density-reachable from a node p with respect to η, if there is a chain of nodes $v_1, v_2, \ldots, v_n$ where $v_1 = p$ and $v_n = q$, such that v_{i+1} and v_i are mutual cores for i ranging from $1, 2, \ldots, n-2$, and v_n is direct density-reachable from v_{n-1}.

Definition 8 (*Density connectivity*) A node q is density-connected to a node p with respect to η, if there exists a node v such that both p and q are density reachable from v.

Density connectivity is a symmetric relation and for the density reachable vertices, it is also reflexive.

Definition 9 (*Density-connected community*) A non-empty subset $C \subseteq V$ is called a density-connected community with respect to η, if all the vertices in C are density-connected with each other and C is maximal with respect to density reachability.

3.2 Overlapping Community Detection

In order to identify overlapping communities in social network data, initially all nodes of the network are un-labeled and un-visited. For a given global percentage threshold, η, the process iteratively finds a density-connected community by randomly selecting an un-visited node, say p, to grow a community using density-reachable relationship of p with other nodes. For each un-visited node p, it checks whether p is a core node and if p qualifies the test, it finds all density-reachable nodes of p to identify its community. To do so, it first computes the local ε_p threshold for p using Eq. 3. If the ε_p threshold for p is greater than zero, then it uses the distance function of Eq. 1 and distance constraint to determine the local ε_p-neighborhood of p, i.e., $N_{local}\,p$. If node p qualifies as a core node, its community list is appended with the current community label and the community list of each node in $N_{local}\,p$ is also appended with the same. We use the term appended as the nodes belonging to $N_{local}\,p$ including p can already be labeled by some other community label(s) in some previous iteration(s). A node is assigned to a new community irrespective of its previous community allotments, thus allowing a node to belong to multiple communities. Once a node p is identified as a core-node, the following important steps are performed for identifying a density-connected community of p.

1. All un-visited mutual-core nodes of node p in $N_{local}\,p$ are appended with the current community label. They are marked as visited and pushed to a stack to identify the density-reachable nodes of p. This step is later repeated for each node in the stack for the current community in order to find the connected sequences of mutual-core nodes starting from p. These connected sequences of mutual-core nodes form the *Mutual-core Connected Maximal Sub-graph* (MCMS) of a community. All nodes in the MCMS of a community are called the primary-core nodes of that community. However, if a core-node p does not show mutual-core relation with any other core-node, then only the node p along with its $N_{local}\,p$ forms a community with p being its only primary core-node.
2. If a core-node q in $N_{local}\,p$ is not a mutual-core of p, it is appended with the current community label; however, it is not pushed into the stack to grow the current community and its visited/un-visited status is kept un-altered.
3. Non-core nodes in $N_{local}\,p$ are marked as visited and they are appended with the current community label. Such nodes form boundary nodes for the community of p and are thus not pushed into the stack as they cannot be used to grow a community.

The steps through 1–3 are repeated for each node in the stack thus identifying a density-connected community for each randomly selected un-visited node p in the social network. It is worthwhile to note that if a core-node q, assigned to a community C, does not show a mutual-core relation with any primary-core node of C, then q is called a secondary-core node of community C and C is called a secondary-community of q. Similarly, if a core-node r is a primary-core node of a community C (i.e., r belongs to the MCMS of C) then community C is called the

primary-community of r. The whole process is repeated for each un-visited node to find the overlapping community structure in the social network. At the end of the process, un-labeled nodes (if any) represent outlier nodes, i.e., they do not belong to any community as they do not show an interaction behavior that is similar to any node or enough number of nodes in the social network.

4 Community Evolutionary Events Tracking

It should be noted that unlike [29], we do not need an ageing function to remove old interactions and we also argue that it is difficult to decide upon a selection criteria to do so. As our approach involves local average interactions of nodes for the clustering process, addition of new interactions results in new averages for the involved nodes and directly effects their neighborhoods and roles for clustering. A social network and its resulting community structure can evolve due to various events triggered by the social network individuals. These events may include:

1. Addition of new weighted interaction links and/or nodes
2. Increase in the interaction weights of existing links
3. Removal of existing nodes

In order to track the evolution of communities in dynamic social networks like OSNs, the proposed method first detects a preliminary community structure from an initial state of the network using the method discussed in Sect. 3.2. Then for each node involved in a change in the network, i.e., the events mentioned earlier, various transitions can occur. They can be handled by either considering a live stream of changes as the network evolves (an online evolutionary adaption of the community structure), or the set of changes observed in a specific time-window (an offline evolutionary adaption of the community structure). In either case, the new edges and/or nodes are added to the network or nodes are removed from the network, and each node involved in a change and its direct-neighbors (nodes with which they have an edge) in the network are marked as un-visited. The reason to consider the direct-neighbors also is that in our proposed method the local ε_p-neighborhood of a node is also dependent on the interaction behavior of its direct-neighbor(s) in a network. So if a node p interacts with some other node q, besides re-determining the local ε_p-neighborhoods of p and q we also need to re-determine the local ε_p-neighborhoods of all the immediate neighbors of p and q respectively to detect the induced change by the nodes p and q. Thereafter, each remaining un-visited node is re-checked for a core-node property by re-calculating its local $\varepsilon_{(p)}$-neighborhood. Various events or transitions used by proposed method to model the evolution of communities are presented in the following sub-sections.

4.1 A Non-core Node Becomes a Core

In this case, either an existing non-core node or a newly added node in the network becomes a core node. In order to track a possible evolutionary event, the following conditions are checked.

For the new core node p, if there exist core nodes in the local $\varepsilon_{(p)}$-neighborhood with which the node p has mutual-core relations and which already belong to different communities, then p causes the primary communities of these core nodes to *merge* into a single community. Consequently, in this case, p causes the MCMSs of different communities to join and form a single MCMS for the new merged community. The merged community also forms the primary community of the new core node p and nodes in its local neighborhood are also added to the merged community.

If the new core node p has mutual-core relations with nodes that have the same primary community C, then p also forms a primary core of community C by appending this community label to itself and to its local neighborhood. This simply results in the *expansion* of community C.

Finally, if the new core node p has no mutual-core relations, then p forms a new community and appends the new community label to its local neighborhood and itself. This causes the *birth* of a new community with p being its only primary core.

4.2 A Core Node Becomes a Non-core

In this case, an existing core node no longer remains a core node due to some change in the network. This triggers either a *split* or a *shrink* event in the evolution of a community as follows.

Let p be a primary core node of a community C at a previous stage, and p seize to exist as a core node due to a new dynamic change in the network. Let S be the set of primary cores of the community C which had mutual-core relations with p before the change in the network. We mark the nodes in S as un-visited. For any core node $q \in S$, let T be a simple BFS (Breadth First Search) traversal of nodes starting from q, visiting nodes in the local neighborhoods of the core nodes and branching at mutual-core relations wherein each newly visited node is labeled as visited. If T includes all the core nodes in S, then p is simply removed from being a primary core of community C. Moreover, if p and/or any other node that belonged to the earlier local neighborhood of p are not in the traversal T, then they are removed with the community label of C, causing C to *shrink*.

However, If T does not include all the core nodes in S, then T forms a new community, i.e., the original community C *split* as p with loosed core-node property causes a cut in the MCMS of C. The community label C of the nodes in T (which now represents a split part of community C) are replaced with a new community label. The traversals are repeated for each remaining un-visited core nodes in S until no further split of community C is possible, i.e., no node in S remains un-visited

after a traversal. In the last traversal, if a node s is visited which does not have the community label of C (i.e., it was removed as s belonged to a previous traversal that split the community C), then the community label of C is re-appended to it resulting in an overlapping node. At the end, the node p and/or any node that belonged to its previous local neighborhood may be labeled with community label C, but do not belong to the last traversal. In this case, the community label C for these nodes is removed, causing community C to further *shrink*.

It is also worth to note that in case a lost core node p was the only primary core node of a community C, then p with loosed core-node property causes the *death* of community C as no representative primary core node for community C remains.

4.3 A Core Node Gains/Looses Nodes but Remains as Core

Due to dynamic nature of social networks, changes in them may cause a core node to gain or loose nodes or both but still hold the core node property. In this case, the addition or removal of nodes are handled as follows.

If the local ε_p-neighborhood of a core node p gains a set of nodes S that do not have *mutual-core* relation with p, then the *primary-community* label of p is simply appended to each node $q \in S$. However, if the added nodes have *mutual-core* relation with p, then they are handled in the same way as the *mutual-cores* of a newly formed core node are handled (Sect. 4.1). This can either cause the *expansion* of a community or *merge* of multiple communities. It is obvious that if all the *mutual-cores* of p in its neighborhood including p have the same *primary-community*, then only the neighborhood of p is updated resulting in *expansion* of a community.

Consider the case when the local ε_p-neighborhood of a core node p with a *primary-community* C, looses a set of nodes L that were earlier in its ε_p-neighborhood. If the nodes in L do not have *mutual-core* relation with p, and they are not direct density-reachable from any other *primary-core* of the community C, then the community label of community C is removed from the lost nodes resulting in the *shrinkage* of community C. However, if a core node p looses a set of nodes S that had *mutual-core* relation with it, then such nodes are handled in the same way when the *mutual-core* of a core node no longer remains a core node (Sect. 4.2). But, in this case the core node p in question is not excluded from the set of nodes S. This could possibly lead to either *split* or no change to the community C.

Most of the community dynamics can be tracked by considering only the three previously mentioned transitions or events and can be used to model the community-centric evolutionary events easily.

5 Parameter (η) Value Estimation

The proposed method requires only a single parameter, η, to be set by the users for detecting overlapping community structures in a social network. The value of η basically defines the size and density of the overlapping communities to be detected.

Smaller values of η yield larger and less-dense communities, whereas larger values yield smaller and more-dense communities. This implies that the parameter, η, can be tuned to detect overlapping community structures at different levels of granularity, naturally forming a hierarchical representation of the identified overlapping communities. In order to find a good approximation for η, the proposed method considers a minimum ($\eta_{\min}$) and a maximum ($\eta_{\max}$) values for η, and the community structures are identified from $\eta_{\min}$ to $\eta_{\max}$ at some regular step until the modularity score [4] of the community structure for the current step is no longer better (or same) than the previous step. In this way, the proposed method takes the value of η between $\eta_{\min}$ and $\eta_{\max}$ as the one where the modularity score of the identified community structure is highest. To define such a domain of η for an underlying network, the proposed method considers the *topological-overlap* (Eq. 6) between a pair (*i, j*) of reciprocating nodes.[1]

$$\sigma_{Overlap} = \frac{|N_i \cap N_j|}{\min(|N_i|, |N_j|)}, \tag{6}$$

In Eq. (6), N_i and N_j represents the sets of nodes to which nodes i and j have out-links, respectively. The *mean* and *standard_deviation* of the *topological-overlap* are taken over all reciprocating pairs of nodes in the underlying network (rounded-up to two decimal places), and the value of *step* is taken as the $standard\_deviation/2$ (rounded-up to two decimal places). The $\eta_{\min}$ value is determined as follows. If $mean + standard\_deviation$ is less than or equal to 0.5, then $\eta_{\min} = mean + step$, otherwise $\eta_{\min} = mean$ (truncated to one decimal place). The $\eta_{\max}$ value is taken as $\eta_{\max} = \eta_{\min} + standard\_deviation$.

The above procedure is used to determine a good approximation of η for every consecutive state of a dynamic network. It is possible that the $\eta_{\min}$ value for a network state at time $t+1$ is less than the η value decided for a previous network state at time t. In this case, all the nodes in the network at time $t+1$ are marked as un-visited and the changes to the local ε-neighborhoods are determined for each node.

6 Overlapping Communities and Post-Merge

As mentioned earlier, the proposed community detection method identifies overlapping community structures in a social network. It does so by allowing a node q to belong to the ε_p-neighborhood of a core-node p irrespective of q's previous community assignments in a density-based context as discussed in Sect. 3.2. Thus a node can belong to multiple communities representing a node where multiple communities overlap. It is often possible that two communities overlap in such a way

[1] For a directed network two nodes are said to be reciprocating if each has an out-going edge towards the other, whereas for un-directed networks each edge is considered to represent a bi-directional reciprocal edge.

that majority of nodes of one community (in some cases both the communities) are involved in the overlap between the two communities. In such cases, two overlapping communities can be merged to represent a single community as implemented in [30]. For the proposed method such a merging of highly overlapping communities is performed as follows. After a community structure is determined from some state of the underlying network at a particular value of η, the proposed method can merge two overlapping communities if the number of nodes, involved in the overlap between them, for the smaller community is more than or equal to the fraction $\eta_{\max}$ of its candidate nodes. In this work, the process of merging highly overlapping communities identified during any state of an underlying network is termed as *post-merge*.

7 Experimental Results

This section presents the experimental results of the proposed method on some benchmark datasets. We compare the results obtained through proposed method with four other state-of-the-art community detection methods that include `MOSES` [24], `DENGRAPH` [23], `gSkeletonClu` [7], and `CFinder` [6]. The evaluation is performed based on two scoring measures which include *omega index* [31] and *normalized mutual information*(NMI) [32]. Both Omega and NMI are generalized scoring measures used for evaluating both overlapping and non-overlapping community structures.

`gSkeletonClu` and `MOSES` are parameter free methods and do not require an input. On the other hand, `CFinder` requires an input parameter k to define the clique size, which has been set to $k = 4$ in our experiment as the method generates best results for this clique size. For `DENGRAPH`, the input parameters ε and μ have been varied to generate the best possible results. All the experiments were performed on an Intel $i3$ based computer with 4 GB memory.

7.1 Results on Static Networks

For this experiment, we have used four well-known real-world benchmarks to evaluate the performance of the proposed method and compared it with other state-of-the-art. For all four real-world network datasets, the ground truth community structures are known and are used to calculate the performance scores. Figure 1 gives the comparison of the proposed method with other state-of-the-art methods on the benchmark datasets.

Figure 1a compares the result scores of the proposed method at $\eta = 62\,\%$ on the un-directed and weighted Zachary's Karate club network [33] with other methods. The proposed method identifies a total of three overlapping communities out of which two almost perfectly match the ground truth. The third community consists of only three nodes out of which one is involved in an overlap with other community resulting

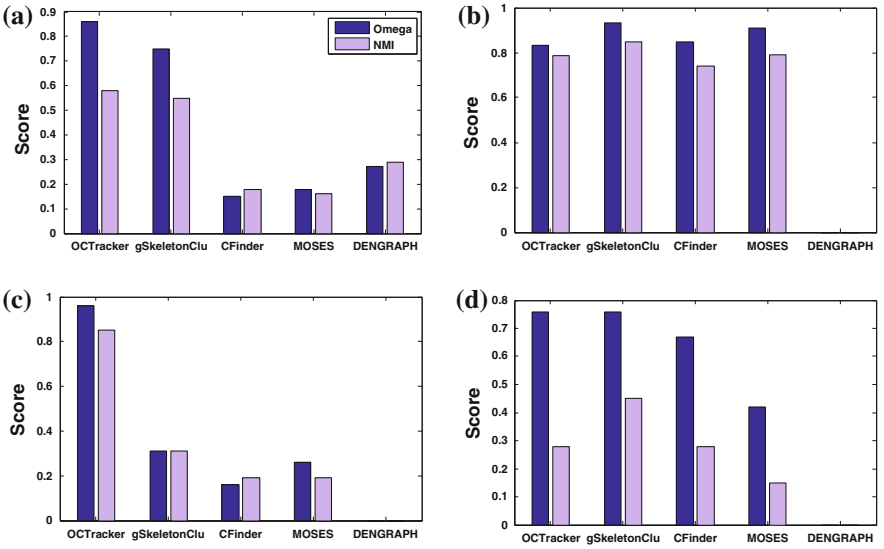


Fig. 1 Experimental results on real-world static datasets. **a** Zachary. **b** Football. **c** Dolphin. **d** Polbooks

in only two misclassified nodes. It can be seen that the community structure identified by the proposed method scores better than all the other methods in question.

Figure 1b gives the comparison of the methods on a 2,000 season NCAA College football network (un-directed and un-weighted) [21], which consists of 115 college football teams, divided into eleven conferences and five independent teams that do not belong to any conference. The proposed method at $\eta = 50\,\%$ exactly identifies eleven communities from the network that almost perfectly match the ground truth. It also identifies five independent teams that do not belong to any conference as outliers. However, it additionally marks three other nodes as outliers and one of the nodes is assigned to two conferences. Figure 1b concludes that almost all methods in question perform well and that the proposed method is comparable to the state-of-the-art methods.

Figure 1c compares the methods on an un-directed and un-weighted social network of frequent associations among 62 Dolphins in a community living off Doubtful Sound, New Zealand that has been compiled by Lusseau et al. [34]. The results obtained by the proposed method are at $\eta = 50\,\%$. It is clear from Fig. 1c that the proposed method performs marginally better that all other methods in question on the Dolphin network.

Figure 1d provides the comparison of the performance scores of the OCTracker with other methods on the US political books network (un-directed and un-weighted). This network is a dataset of books about US politics compiled by Valdis Krebs http://www.orgnet.com/ wherein the nodes represent books about US politics sold online by Amazon and the edges represent frequent co-purchasing of books by the same buyers.

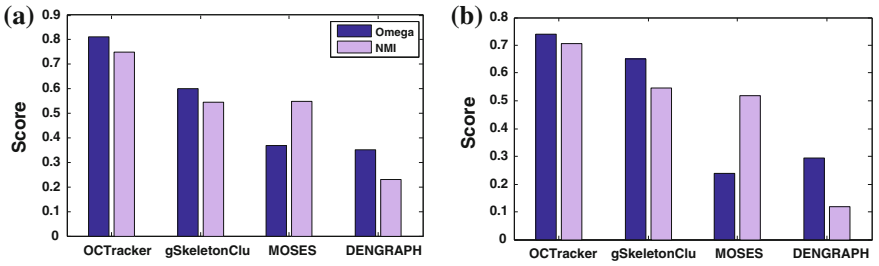


Fig. 2 Experimental results on a primary school dynamic network considering the network **a** after day-1, and **b** after day-2 (i.e., merged day-1 and day-2 network)

Mark Newman http://www-personal.umich.edu/~mejn/netdata/ clustered the nodes of this network into 'liberal', 'neutral' and 'conservative' based on the description and reviews of books posted on Amazon. The network consists of 105 nodes (books) and 441 edges (co-purchases). The proposed method identifies a total of five communities at $\eta = 62\,\%$ with four overlapping nodes and two outliers. Two of the identified communities closely match to the actual 'liberal' and 'conservative' categories. However, the 'neutral' category is difficult to identify and is scattered into three communities by the proposed method along with a few nodes from the 'liberal' and 'conservative' categories. Figure 1d shows that the proposed method also performs reasonably well on the political books network dataset. It is notable from Fig. 1 that DENGRAPH [23] is not able to identify the community structure in un-weighted networks.

7.2 Results on Dynamic Networks

This section presents experimental results on two dynamic network datasets. The first dataset comprises two weighted networks of face-to-face proximity between 242 individuals representing students and teachers in a primary school over a period of two days [35]. The two networks correspond to two days of study wherein a daily contact network is provided. The nodes in this network represent students and teachers, and edges correspond to the interactions between them. The weight of an edge represents the number of times two nodes have interacted during the day. The students actually belong to ten different classes which can represent the ground truth communities. The teachers do not specifically belong to any class and interact with any student community. Our aim is to track the community-centric evolutionary events that possibly occur during the two days of interactions between the individuals. We also aim to see how well can the actual communities in the network, at various stages, be detected by the community detection methods.

Figure 2a shows the comparison of performance scores (Omega and NMI) for the various methods on the interaction network of the individuals after day-1. The scores

are computed against the known ground truth for day-1. As can be seen from the results for day-1, the proposed method performs better than all other methods in question.

In order to detect the evolutionary events, the set of community structures detected by the proposed method for day-1 forms its initial community state. This initial community structure is now adapted to changes in the network, i.e., by adding interactions for day-2 to the underlying network which could also include adding new nodes, as discussed in Sect. 4. Figure 3 shows the dynamic changes that occur in the community structure of the primary school interaction network over two days as tracked by the proposed method. Initially on day-1 network, the proposed method detects 9 communities labeled as *A*–*H*, of which community *C* overlaps with *D* and *E* overlaps with *I*. The interactions for day-2 are merged with the underlying day-1 network which leads to addition of some new nodes and edges, and increases the weights of some already existing edges. Thereafter, OCTracker scans the changes in the network as discussed in Sect. 4 and tracks the resulting community-centric changes in the initial community structure. As shown in Fig. 3, almost all the initial communities gain nodes resulting in their expansion. Two important evolutionary events are detected by the proposed method after the second day of interactions. Firstly, the two overlapping communities *C* and *D* merge to form a single community labeled as $C + D$. Secondly, community *G* splits into two overlapping communities labeled as G_1 and G_2. Moreover, after the second day of interactions, many communities[2] begin to overlap with each other which are represented by overlapping circles in Fig. 3.

Figure 2b shows the comparison of performance scores (Omega and NMI) for the various methods on the interaction network of the individuals after day-2, i.e., the network represented by merging the interactions and nodes for both day-1 and day-2. The scores are computed against the known ground truth for both day-1 and day-2 data. As can be seen from the results, the proposed method again performs better than all other methods in question for the complete primary school interaction network over two days. To generate the results for the proposed method on the primary school network dataset, the input parameter η is set to 65 %. Surprisingly, `CFinder` could not generate any results for the primary school network data due to its higher space complexity.

The second dataset [36] is a dynamic directed-network of about 8,000 users from the English Wikipedia that voted for and against each other in admin elections from year 2004 to 2008. Nodes represent individual users, and directed-edges represent votes. Edges are positive ("for" vote) and negative ("against" vote) represented by edge weights of 1 and -1, respectively. For this paper, the dataset is divided into five subnetworks based on the year of voting. Starting with the network of year 2004, the proposed method identifies the preliminary community structures. Then for each subsequent year, it adds the respective subnetwork to the current state of the network

[2] Figure 3 does not depict the actual size of the detected communities or the amount of overlap between communities.

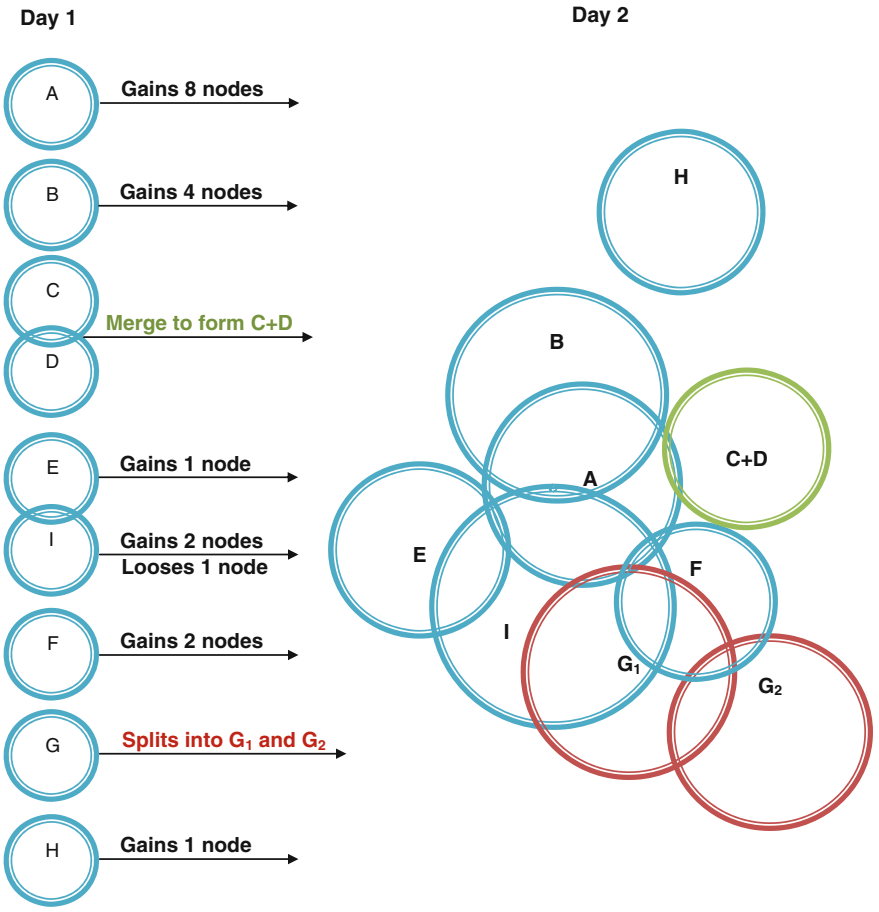


Fig. 3 Community evolution tracking in a primary school dynamic network

and identifies the changes induced to the existing community structures for the new state of the network. The proposed method finds highly-overlapping communities from each state (cumulative network from the start to some later year) of the voting network. Some of the evolutionary transitions (birth, split, and merge) for some of the communities across any two consecutive states (years) of the voting network identified by the proposed method (without post-merge) is shown in Fig. 4. Based on these results, we conclude that the proposed method can identify the evolutionary transitions (birth, growth, contraction, merge, split, and death) of communities across a time-varying network even if the changes involve only the addition of new edges and/or nodes. It means that the proposed method does not necessarily require an ageing function to remove old links.

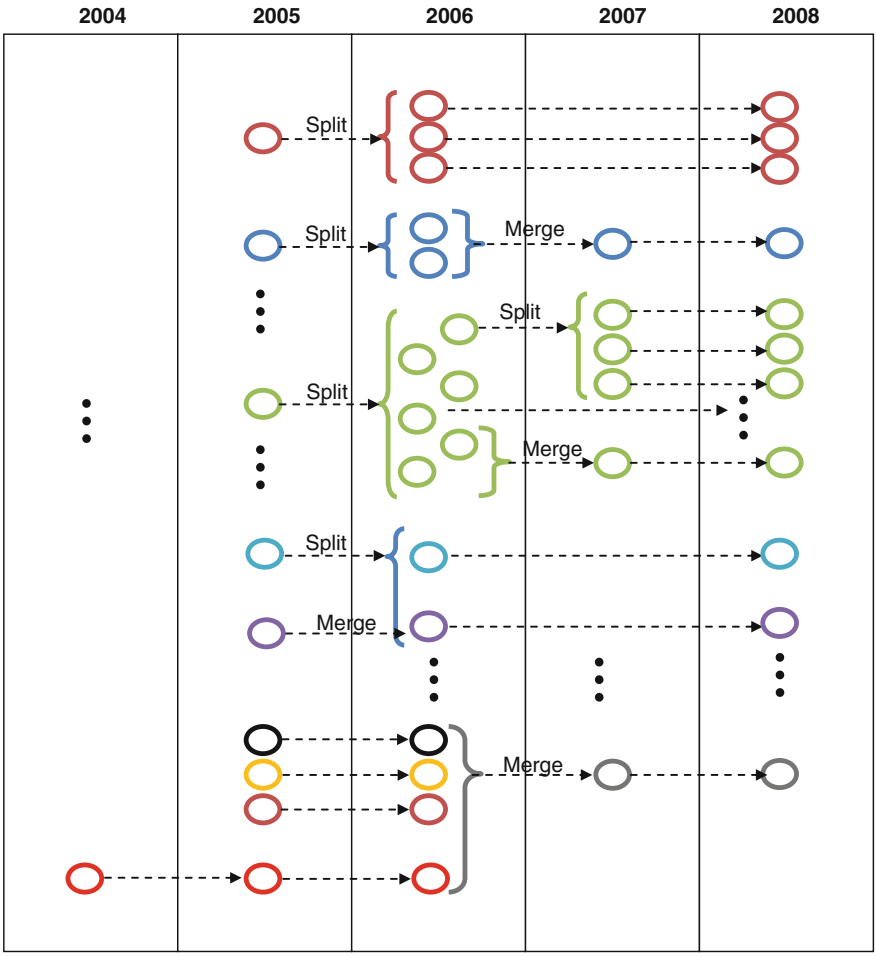


Fig. 4 Community tracking on the signed and directed Wikipedia election network [36]

As mentioned earlier, the partial results on the Wikipedia election network shown in Fig. 4 are generated by the proposed method without performing the post-merge process. On applying post-merge to the community structure identified for each state of the network, the number of communities for each state are reduced as many highly overlapping communities are merged to represent a single community. The analysis of the community evolution trend, using post-merge with the proposed method, reveals that at every new state new nodes tend to join existing larger communities (and cause their growth) or form completely new communities instead of involving in merge or split.

8 Conclusion

This chapter has presented a novel density-based approach to track the evolution of overlapping community structures in online social networks. The novelty of the proposed method lies in the approach for allowing the communities to overlap, and its distance function which is defined as a function of the average interactions between a node and its neighborhood. In addition, unlike other density-based methods for which the neighborhood threshold is to be set by the users, which is generally difficult to determine, the proposed method computes a local neighborhood threshold for each node from the underlying network. The preliminary experimental results on both static and dynamic networks show that the proposed method is comparable to the state-of-the-art methods and can effectively track the evolutionary events in dynamic networks. The method is naturally scalable to large social networks.

References

1. Ding CH, He X, Zha H, Gu M, Simon HD (2001) In: Proceedings of the international conference on data mining, pp 107–114
2. White S, Smyth P (2005) In: Proceedings of the 5th SIAM international conference on data mining, pp 76–84
3. MacQueen JB (1967) In: Proceedings of the 5th Berkeley symposium on mathematical statistics and probability, vol 1. University of California Press, California, pp 281–297
4. Newman ME, Girvan M (2004) Finding and evaluating community structure in networks. Phys Rev E 69
5. Handcock MS, Rafter AE, Tantrum JM (2007) Model-based clustering for social networks. J Royal Stat Soc A 170:301
6. Palla G, Derényi I, Farkas I, Vicsek T (2005) Uncovering the overlapping community structure of complex networks in nature and society. Nature 435(7043):814
7. Sun H, Huang J, Han J, Deng H, Zhao P, Feng B (2010) In: Proceedings of the IEEE international conference on data mining, ICDM '10. IEEE Computer Society, Washington DC, pp 481–490
8. Kim MS, Han J (2009) In: Proceedings of the 12th international conference on discovery science, DS '09. Springer, Berlin, Heidelberg, pp 152–167
9. Bhat SY, Abulaish M (2012) In: Proceedings of the international conference on web intelligence, mining and semantics (WIMS'12). ACM, pp 9:1–9:7
10. Ester M, Kriegel H, Jörg S, Xu X (1996) In: Proceedings of the international conference on knowledge discovery from data, pp 226–231
11. Tantipathananandh C, Berger-Wolf T, Kempe D (2007) In: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '07. ACM, New York, pp 717–726
12. Lin YR, Chi Y, Zhu S, Sundaram H, Tseng BL (2009) Analyzing communities and their evolutions in dynamic social networks. ACM Trans. Knowl. Discov. Data 3, 8:1
13. Wilson C, Boe B, Sala A, Puttaswami KP, Zhao BY (2009) In: Proceedings of the 4th ACM European conference on computer systems. ACM, New York, pp 205–218
14. Chun H, Kwak H, Eom Y, Ahn Y, Moon S, Jeong H (2008) In: Proceedings of the 8th ACM SIGCOMM conference on internet measurement, vol 5247, pp 57–70
15. Bhat SY, Abulaish M (2012) In: Proceedings of the IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM'12). IEEE Computer Society

16. Kernighan BW, Lin S (1970) An efficient heuristic procedure for partitioning graphs. Bell Syst Tech J 49:291
17. Bezdek JC (1981) Pattern recognition with fuzzy objective function algorithms. Kluwer Academic Publishers, Norwell
18. MacQueen JB (1967) In: Cam LML, Neyman J (eds) Proceedings of the 5th Berkeley symposium on mathematical statistics and probability, vol 1. University of California Press, California, pp 281–297
19. Scott JP (2000) Social network analysis: a handbook, 2nd edn. Sage Publications Ltd., California
20. Wasserman S, Faust K (1994) Social network analysis: methods and applications. Cambridge University Press, Cambridge
21. Girvan M, Newman ME (2002) In: Proceedings of the national academy of sciences, vol 99, pp 7821–7826
22. Xu X, Yuruk N, Feng Z, Schweiger TAJ (2007) In: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining (KDD '07). ACM, pp 824–833
23. Falkowski T, Barth A, Spiliopoulou M (2007) In: Proceedings of the IEEE/WIC/ACM international conference on web intelligence. IEEE Computer Society, Washington DC, pp 112–115
24. McDaid A, Hurley N (2010) In: Proceedings of the 2010 international conference on advances in social networks analysis and mining, ASONAM'10. IEEE Computer Society, Washington DC, pp 112–119
25. Latouche P, Birmel E, Ambroise C (2009) Bernoulli in press(25), 1
26. Backstrom L, Huttenlocher D, Kleinberg J, Lan X (2006) In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '06. ACM, New York, pp 44–54
27. Palla G, lászló A, Barabási T, Vicsek B (2007) Hungary. Nature 446:664
28. Greene D, Doyle D, Cunningham P (2010) In: Proceedings of the 2010 international conference on advances in social networks analysis and mining, ASONAM '10. IEEE Computer Society, Washington DC, pp 176–183
29. Falkowski T, Barth A, Spiliopoulou M (2008) In: Proceedings of the 14th Americas conference on information systems (AMCIS)
30. Dourisboure Y, Geraci F, Pellegrini M (2007) In: Proceedings of the 16th international conference on world wide web, WWW '07. ACM, New York, pp 461–470
31. Collins LM, Dent CW (1988) Omega: A General Formulation of the Rand Index of Cluster Recovery Suitable for Non-disjoint Solutions. Multivar Behav Res 23(2):231
32. Lancichinetti A, Fortunato S, Kertész J (2009) Detecting the overlapping and hierarchical community structure in complex networks. New J Phys 11:033015
33. Zachary WW (1977) An information flow model for conflict and fission in small groups. J Anthropol Res 33:452
34. Lusseau D, Schneider K, Boisseau OJ, Haase P, Slooten E, Dawson SM (2003) The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations. Behav Ecol Sociobiol 54:396
35. Stehl J, Voirin N, Barrat A, Cattuto C, Isella L, Pinton JF, Quaggiotto M, den Broeck WV, Rgis C, Lina B, Vanhems P (2011) CoRR abs/1109.1015
36. Leskovec J, Huttenlocher D, Kleinberg J (2010) In: Proceedings of the 19th international conference on world wide web, WWW '10. ACM, New York, pp 641–650. DOI10.1145/1772690.1772756

@Rank: Personalized Centrality Measure for Email Communication Networks

Paweł Lubarski and Mikołaj Morzy

Abstract Email communication patterns have been long used to derive the underlying social network structure. By looking at who is talking to who and how often, the researchers have disclosed interesting patterns, hinting on social roles and importance of actors in such networks. Email communication analysis has been previously applied to discovering cliques and fraudulent activities (e.g. the Enron email network), to observe information dissemination patterns, and to identify key players in communication networks. In this chapter we are using a large dataset of email communication within a constrained community to discover the importance of actors in the underlying network as perceived independently by each actor. We base our method on a simple notion of implicit importance: people are more likely to quickly respond to emails sent by people whom they perceive as important. We propose several methods for building the social network from the email communication data and we introduce various weighting schemes which correspond to different perceptions of importance. We compare the rankings to observe the stability of our method. We also compare the results with an *a priori* assessment of actors' importance to verify our method. The resulting ranking can be used both in the aggregated form as a global centrality measure, as well as personalized ranking that reflects individual perception of other actors' importance.

P. Lubarski (✉) · M. Morzy
Institute of Computing Science, Poznan University of Technology, Piotrowo 2,
60-965 Poznan, Poland
e-mail: Pawel.Lubarski@cs.put.poznan.pl

M. Morzy
e-mail: Mikolaj.Morzy@put.poznan.pl

F. Can et al. (eds.), *State of the Art Applications of Social Network Analysis*,
Lecture Notes in Social Networks, DOI: 10.1007/978-3-319-05912-9_10,

1 Introduction

Can we establish the importance of people by simply analyzing the database of sent and received emails, having no access to subject lines or contents of messages? The answer, apparently, is "yes we can". Intrinsic behavior of people reveals simple patterns in choosing which email to answer next. Our theory is based on two assumptions. We assume that people do their email communication in bursts, answering several messages consecutively and that they can freely choose the order of answers. Secondly, we believe that people use priority queues to manage their internal task lists, including the list of emails to be answered. Looking at timing and ordering of responses we derive individual rankings of importance, because we posit that people have a tendency to reply to important actors first. These individual subjective rankings are significant because they reflect the relative importance of other actors as perceived by each actor. The individual rankings can be further aggregated into a global ranking of importance of all actors, which can be used as a centrality measure.

Contemporary communication within institutions and organizations in being performed mostly using emails. In many cases this is the preferred method of communication, strongly favored over other communication modalities. Emails allow people to pass information quickly without having to physically contact each other. Due to the overwhelming usage of emails and the sheer volume of communication, both professional and private, many researchers have begun to treat email not as a pure communication tool, but rather as a tool for prioritizing goals and tasks (where one's inbox serves as the to-do list of tasks). This approach to email tools allows us to use email communication to study human behavior and social interactions.

Our theory is strongly influenced by ideas of Albert László-Barabási. In his latest book, "Bursts" [4], he formulates an interesting proposition. According to him, human behavior is mostly controlled by priority queues that humans create and manage internally (and often unconsciously). For instance, a person has a priority queue for handling email communication. The length of the queue may vary from person to person and may depend on individual traits, but psychological experiments suggest that the average queue length should not exceed 10 items. The arrival of emails is dictated by the specifics of person's profession, so without a loss of generality we may assume that the arrival times of emails follow the Poisson distribution. Given a set of current emails in the queue (where each email represents a task), László-Barabási theorizes that the position of each email in the queue corresponds to the perceived importance of the task. If a new email arrives at a given moment, the person evaluates the importance of the email (in the context of the current email queue) and places the email in the queue at the appropriate position, pushing less important emails down the queue. Of course, this process is unconscious and we may only observe its outcome. Interestingly, if applied to the domain of email communication, this scenario produces a power law distribution of email response times that is very similar to email response distributions observed in real datasets. According to László-Barabási, the phenomenon of priority lists management can be applied

to many different aspects of human activity and can account for various power law distributions observed in practice.

We have decided to test this model and apply it to the large body of email communication data obtained from the Institute of Computing Science at Poznan University of Technology. We assume that every person within the Institute is familiar with other people to the extent that allows her to assess the relative professional importance of each person. If the model is correct, we should be able to derive the ranking of employees that corresponds to the perceived importance of employees by using the communication data and constructing the network, where each node represents an employee and each edge represents an email conversation. Conversations are weighted based on the behavioral characteristics of people involved. The weight may represent the delay in answering, the number of other email messages in the queue that were pushed down by the current conversation, or simply the number of emails in the queue at the moment of conversation. Based on these weighting schemes we develop several local rankings that order the employees. Next, we aggregate these rankings and compute distances between rankings.

In particular, we seek to answer the following questions:

- Can we compute efficiently the ranking of employees?
- Is there a significant difference in rankings by various weighting schemes?
- To which extend our ranking resembles the ranking computed solely from employees' university positions (research assistants, associate professors, etc.)?
- Do our rankings reveal interesting patterns of perceived importance?
- Do local priority queues derived from individual employees aggregate to meaningful global rankings?

We perform an experimental evaluation of the model by analyzing the dataset consisting of over 600 000 emails sent during one year period to 200 employees of our university. The results of our experiments show that there is a strong correlation between the perceived importance of people and the time taken to answer emails. Indeed, we have the tendency to put emails from important people on top of the priority queue. Extending individual perception of importance by the information from other employees allows us to construct the social network which closely models the relationships of importance between the employees. The resulting global ranking correctly identifies tenure professors and functional employees (vice-director of the Institute, finance director of the Institute) at the top of the ranking. At the same time, very interesting patterns emerge (such as a very high position of a person who served as the local coordinating officer for a large European project, at the same time being officially only a regular assistant professor). We firmly believe that the presented method can be employed to other institutions to unearth hidden patterns and implicit importance of employees.

We think that our model is general and it can be applied whenever behavioral data is available which includes any choice made by actors from a set of available alternatives with the alternatives having varying degrees of importance to individual actors. The method is computationally inexpensive and requires no prior knowledge before computing the social network. It can be easily applied in real time and can be

used both on the server side (e.g., as a part of an anti-spam filter) and on the client side (e.g., by enhancing the functionality of email client software). The ranking of users computed by our method can be used to provide additional ordering criterion to boost productivity, to mark urgent emails that are procrastinating in the inbox, or to provide additional insights into correspondent's importance. At the global level, the analysis of the resulting rankings reveals the "true" importance of actors as perceived by the community, which, in turn, allows to identify unrecognized influential employees as well as high profile employees who are deemed unimportant by others. The results can also be useful in evaluating customer relations and assessing the impact of marketing campaigns.

The presented method is not without its limitations. It operates under the assumption that email communication is the primary communication means within the organization. Though it is often the case, individuals may exist (in particular, on the highest levels of organizational hierarchy) who do not employ email as communication means. Such individuals often relegate email communication to subordinate staff and do not communicate directly using email. We have observed this phenomenon in our dataset, where the importance of the head of the Institute was clearly transfered to the secretary, who serves as a proxy. In such cases, additional configuration is required to capture such cases in the data.

The organization of the chapter is the following. In Sect. 2 we present the related work on the subject. Section 3 introduces basic definitions and notation used throughout the chapter. In Sect. 4 we report on the results of the initial experimental evaluation of the model. We present our extension of the model in Sect. 5 and we conclude the chapter in Sect. 6 with a brief summary.

2 Related Work

The literature on email communication analysis is abundant. Probably, the most famous analysis has been conducted on email communication within the Enron group shortly before the outbreak of the scandal [11]. For instance, using the Enron dataset Trier and Bobrik [21] develop a new centrality measure called Brokering Activity that captures the involvement of each node in the formation of the network and analyzes the growth of the network using overlapping time windows. An overview of mining email social networks can be found in [7]. An interesting system for extracting user social network by analyzing the contents of user's inbox is presented in [10]. Also, previous works exist that aim at analyzing email communication networks and using reputation of users to fight spam [14].

In many aspects the analysis of email communication data resembles the analysis of the Twitter environment. The similarities stem from the fact that Twitter conversations also employ the threading of tweets and replies. More importantly, the user is presented with an "inbox" of recent tweets and may freely choose the order of replies, which is crucial for our model. The research on the characteristics of the Twitter abounds. In [8] authors analyze the characteristics of conversations and

show how the new mechanism of retweeting is being employed in many different ways to conduct conversations between individuals, groups and public. Most of the research on Twitter networks concentrates however on finding the most influential users (which is somehow similar to our quest of finding the most respected and reputable employees). Authors of [1] argue that although one may define "power-users" of Twitter as active users with many followers who produce cascades of retweets, the prediction whether a particular resource or a particular tweet will be retweeted are very unreliable and that the only way to induce viral effects in the Twitter network is to target a large number of users in order to harness the average effect. Similar conclusions can be drawn from [9] where authors find that simple centrality measures such as indegree are poor predictors of one's influence. In [22] a new centrality measure, called *TwitterRank*, is introduced and the authors claim that this measure outperforms other approaches to the quantification of user's influence.

Our current work is mostly influenced by the works of László-Barabási, in particular, by [4–6]. We employ the idea presented in [4] and verify it on a real dataset, adding novel weighting schemes and comparing various rankings, but the main premise of the research presented in this chapter is derived from these works. A similar approach is presented in [15], but the authors there concentrate on the design of a universal priority queue model to explain the output power law distributions of events with both Poisson and power law distributions of input stimuli.

We have presented our initial findings in the area of using email communication networks to build personalized rankings of actor's importance in [17]. In this chapter we present the overview of our method and we extend our previous findings by filtering out quick interruptions and incorporating the measurement of the effort required to prepare a message. The detailed description of the new material w.r.t. the original paper is presented in Sect. 5.

In this chapter we also use rank aggregation and comparison methods. Ranking of elements and the aggregation of partially ranked lists has long been the subject of scientific inquiry. A thorough introduction to rank aggregation methods can be found in [12]. Most of the research has been conducted on ranking search engine query results, for instance [2, 3, 13]. Also, a significant effort has been put into developing effective aggregation methods for full and partial ranking lists [16].

Our work also touches upon reputation management. This area of research has always attracted lot of scientific attention. For a quick introduction we refer the reader to [18–20].

3 Basic Definitions

In this section we introduce basic definitions used throughout the chapter and we present various weighting schemes employed for the construction of the underlying social network.

Let $U = \{u_1, \ldots, u_m\}$ be the set of users. Let m_{ij}^t denote the fact of sending a message from user u_i to user u_j at time t. Let M be the set of all sent messages.

If $m_{ij}^{t_0} \in M \wedge m_{ji}^{t_1} \in M$, we say that there was a *communication* between users u_i and u_j. If $t_0 < t_1$ then the user u_j replied to the message sent by the user u_i, which is denoted as $c_{ji}^{t_1}$. In practice, users u_i and u_j might have communicated several times over different issues. Due to the privacy concerns we did not have the access either to subject lines of emails or to the `In-Reply-To` header field, so we could not have identified multiple instances of communication between users u_i and u_j as concerning different topics. Therefore, we make a simplifying assumption that users u_i and u_j communicate in a serial mode using the first-in-first-out method, i.e., the first email from u_i to u_j is answered by the first email from u_j to u_i, the second email from u_i to u_j is answered by the second email from u_j to u_i, and so on. In case if there are multiple emails from u_i to u_j before the first response, i.e. if $m_{ij}^{t_0}, \ldots, m_{ij}^{t_n} \in M$ then the first message $m_{ji}^{t_{n+1}}$ from u_j to u_i replies to the entire set of $m_{ij}^{t_0}, \ldots, m_{ij}^{t_n}$ and is denoted by $c_{ji}^{t_n+1}$.

Every reply has a delay $\tau(c_{ji}^{t_k}) = |t_k - t_{k-1}|$. With each user we associate a *message queue* which represents all messages received by the user until time t. This message queue is denoted as $Q^t(u_i)$.

$$Q^t(u_i) = \{m_{ki}^{t'} \in M : t' < t, u_k, u_i \in U\}$$

After being replied the message is discarded from the queue. Thus, if there is a message $m_{ik}^{t''} \in M : t'' > t'$ then the message $m_{ki}^{t'}$ is removed from the message queue. The number of messages received during the communication delay $\tau(c_{ji}^{t_1})$ is called the *delay queue* and it is denoted as $q(c_{ji}^{t_k}) = \text{abs}\left(\left|Q_{u_i}^{t_{k-1}}\right| - \left|Q_{u_i}^{t_k}\right|\right)$.

Let $R(u_i)$ represent the centrality rank of the user u_i. The weight of the communication between users u_i and u_j is denoted as w_{ij}. We may perceive the weight w_{ij} as the importance of the user u_j as perceived by the user u_i. In other words, it is a vote from u_i to u_j. In general, $w_{ij} \neq w_{ji}$. Independent of a particular weighting function, we compute the rank of the user u_i recursively by computing the ranks of all users who communicate with u_i and considering the weights of these communications:

$$R(u_i) = \sum_{u_k \in U} w_{ki} * R(u_k) \tag{1}$$

Because there may be several instances of communication between a given pair of users, each with a different weight, we have taken an additional simplifying assumption that the final weight w_{ij} is averaged over all communications between the given pair of users:

$$w_{ij} = \sum_{c_{ij} \in C} \frac{w(c_{ij}^t)}{\left|\{c_{ij} \in C\}\right|} \tag{2}$$

In our experiments we have considered the following three weighting schemes for communication

3.1 Delay Factor

Our first scheme weights the response by the time it took to answer, decreasing the weight linearly with the time. The weighting is given in Eq. 3:

$$w_{ij}^{DF} = \begin{cases} 1 & \text{if } \tau(c_{ij}^t) < \alpha \\ \frac{1}{\tau(c_{ij}^t) - \alpha} & \text{otherwise} \end{cases} \tag{3}$$

where α denotes the single time unit duration, i.e., the interval within which the typical response is expected to happen. This time unit strongly depends on the means of communication. For instant messaging this would be close to a couple of minutes, for email this may extend to a couple of hours, whereas for Internet forum posts this may extend to several days. In our experiments we have set $\alpha = 8$, arbitrarily choosing the delay of less than 8 h to be acceptable. In other words, if an email has been answered within 8 h from the moment of arrival, we assume that it has been answered "instantly", without unnecessary delay. The rationale behind such choice is that people usually work in day batches and it is expected that email messages will be answered within a single working day.

3.2 Queue Number

The second scheme weights the response by the number of messages in the queue that were answered after the response. In other words, the highest weight is always given to the first answered message. In this scheme we fix the size of the message queue at β messages and the weight of the response is β if this has been the first message the user has answered, $\beta - 1$ if this has been the second message the user has answered, and so on. Equation 4 presents the formula for the weighting function:

$$w_{ij}^{QN} = \begin{cases} \beta - q(c_{ij}^t) & \text{if } q(c_{ij}^t) <= \beta \\ 0 & \text{if } q(c_{ij}^t) > \beta \end{cases} \tag{4}$$

where β denotes the *a priori* chosen length of the priority queue. Psychological research suggests that the lesser number of choices increases the level of human happiness and that too many choices are derogatory for our cognitive and processing abilities. It is suggested that typical to–do lists should not exceed 8–10 items. However, since we are dealing with homogeneous tasks (email answering) with a great variance in labor intensiveness between them, we have decided to set the length of the queue at $\beta = 15$ emails. This choice is arbitrary and not backed by any hard evidence. However, we have tried a few settings of the β parameter and we have not observed any significant change to the resulting rankings.

3.3 Queue Factor

In the third weighting scheme we compute, for each message answered, the number of preceding unanswered messages in the queue divided by the size of the queue. In other words, for each message that has been answered we compute the number of messages that were shifted down the queue by the current message. This scheme presented in Eq. 5 aims at identifying as important the conversations which interrupt the priority queue of the user.

$$w_{ij}^{QF} = \frac{\left|Q_{u_i}^t\right| - q(c_{ij}^t)}{\left|Q_{u_i}^t\right|} \tag{5}$$

3.4 Global Versus Local Ranking

For each of the above weighting schemes we have run the ranking algorithm in two versions: with and without user weighting. The two rank definitions were as follows:

$$R(u_i) = \sum_{u_k \in U} w_{ki} \tag{6}$$

$$R(u_i) = \sum_{u_k \in U} w_{ki} * R(u_k) \tag{7}$$

Equation 6 presents the simple version of the ranking function where the rank of the user is determined solely based on the weights of communication edges, without considering the ranks of users with whom the given user communicates. An extended formula, presented in Eq. 7, considers also the current ranking of the user with whom the given user communicates (we will refer to these schemes as *weighted* schemes). It is worth noting that this formulation of the ranking problem produces a single global ranking of users, thus leading to a global centrality measure of user's importance as averaged over all other users. On the other hand, one might decide to create vectors $R_L(u_i) = [w_{1i}, w_{2i}, \ldots, w_{mi}]$ representing, for each user u_i, the relative importance of users $u_1, u_2, \ldots, u_m$ to the user u_i. In other words, our method produces both a global ranking $R(u_i)$ or users and local ranking $R_L(u_i)$ personalized for each user u_i. In the next section we report on the results of the experimental evaluation of our method.

4 Experiments

The data consisted of one year worth of email communication within the Institute of Computing Science at Poznan University of Technology. The dataset describes the communication between 126 224 actors sending 637 284 email messages (the

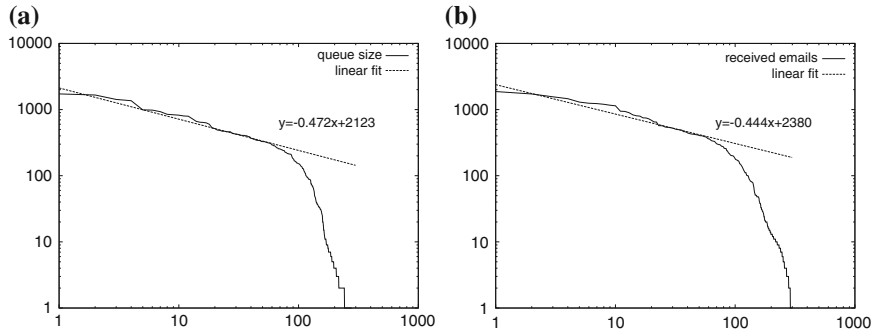


Fig. 1 Queue length histogram

majority of actors are external entities and only around 200 people are employed in the Institute). The dataset has been sanitized by removing spam messages, messages generated by software (e.g., SVN reports), aggregating messages to/from the same actor using different university aliases, and removing emails with an empty sender header. In our experiments we have assumed the basic unit of time to be 1 h.

Figure 1a, b present histograms of key variables of the dataset. As one can easily see, the distributions of received messages, as well as the distribution of the message queue follow power law distributions with exponential cut-offs (the distribution of sent messages is almost identical to the distribution in Fig. 1b). These cut-offs roughly correspond to physical limits of email communication. For instance, there are very few employees who have a queue of unanswered emails longer than 100 messages.

Next, we have run our ranking algorithms on the networks induced by the weighting schemes described in Sect. 3. As the result, we have received several different rankings. Firstly, we have decided to verify the quality of rankings by performing a simple comparison. We have manually created a list of all employees and we have mapped each employee to a particular position within the Institute (these positions basically represent different jobs and titles, such as Ph.D., research assistant, associate professor, tenure professor, etc.). Each position was then assigned a certain amount of points proportionally to its "importance". For instance, tenure professors were assigned 80 points, while Ph.D. students were assigned 10 points. People from outside of the Institute were assigned 1 point. As of the time of the experiment the Institute counted 188 employees. In addition, employees who were occupying administrative positions in the Institute were awarded additional 20 points for their position (e.g., vice-deans, financial director, etc.).

Given a ranking of employees, we have divided the employees into 10-element bins and we have aggregated the points in each bin. We present these moving partial sums in each bin for selected ranking algorithms in Fig. 2a through c. One may see that there are no apparent differences between rankings and each ranking produces almost strictly monotonically decreasing partial sums. The variation in sums of points across bins may be easily explained by the fact that some employees, who are formally

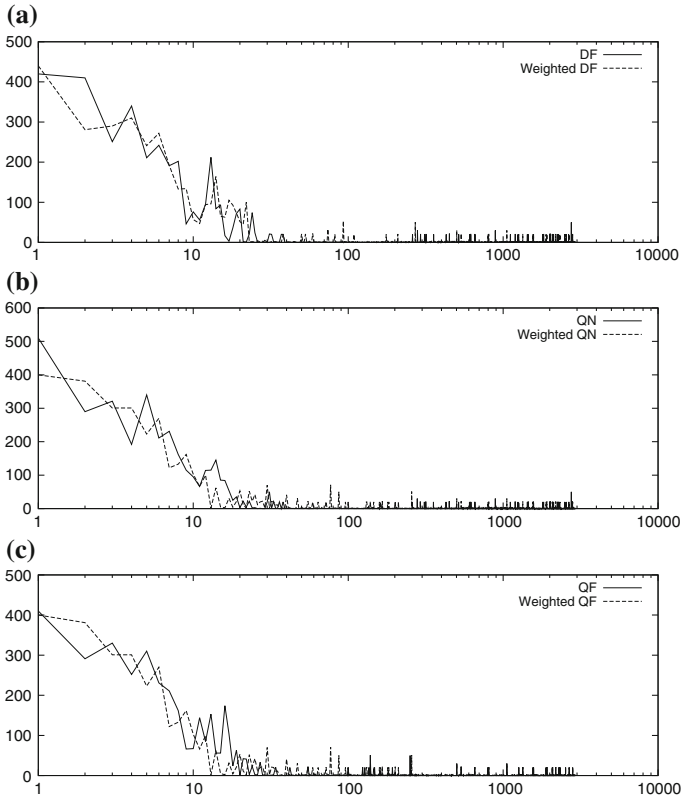


Fig. 2 Delay factor weighting function ranking

"important" (e.g., employees who are tenure professors), may not be that important in the eyes of other employees. Indeed, when scrutinizing ranking lists of employees we have discovered, that each ranking correctly identified the most important people in the Institute (for instance, each ranking had two members of the Academy of Sciences ranked within the first 5 positions). In addition, each algorithm ranked very highly the secretary of the head of the Institute (while the head of the Institute was ranked low, because this person uses the proxy to communicate with subordinates). Each algorithm correctly identified all automatically generated communication and assigned the weight of such communication to 0.

Because we were not sure whether any of the weighting schemes has proven more advantageous than others, we have computed distance measures between resulting rankings to see the variability of ranking results. To compare rankings we have used well known measures of Spearman's ρ footrule distance and Kendall's τ distance.

Given a universe Ψ, an ordered list is a ranking $r = [x_1 \geq x_2 \geq \ldots \geq x_n]$ with each $x_i \in \Psi$ and $\geq$ being some ordering relation on Ψ. The Spearman's ρ footrule distance is the sum of the absolute differences between the ranks of x_i over all rankings,

Table 1 Similarity between rankings shown by Spearman's ρ and Kendall's τ

	DF	Weighted DF	QN	Weighted QN	QF	Weighted QF
DF	1.00	0.85	1.00	0.87	0.85	0.85
Weighted DF	0.85	1.00	0.85	0.88	0.90	0.90
QN	1.00	0.85	1.00	0.87	0.85	0.85
Weighted QN	0.87	0.88	0.87	1.00	0.88	0.88
QF	0.85	0.90	0.85	0.88	1.00	0.90
Weighted QF	0.85	0.90	0.85	0.88	0.90	1.00

for all items $x_i \in \Psi$. Formally, given rankings r_1 and r_2 the Spearman's footrule distance between these rankings is defined as $DFD(r_1, r_2) = \sum_{i=1}^{|\Psi|} |r_1(i) - r_2(i)|$. After dividing this value by the maximum value of $|\Psi|^2/2$ we obtain the normalized Spearman's footrule distance. Analogously, the Kendall's τ distance is defined as the number of pairwise disagreements between two lists. Sometimes the Kendall's τ distance is called the bubble sort distance because it captures the same aspect of the data. Formally, $KTD(r_1, r_2) = |\{(i, j) : i < j, r_1(i) < r_2(i) \wedge r_1(j) > r_2(j)\}|$. Table 1 summarizes the correlation similarities between different rankings (where similarity has been computed directly from the distance). We report a single value because the differences between the two ranking distances would not appear for less than three digits of precision. Therefore, for the sake of simplicity we conclude that the two distance measures produce the same result. The outcome presented in Table 1 is very promising because high mutual similarity between rankings clearly shows that all weighting functions are valid and that they capture a very similar set of data features.

5 Extended Work

5.1 Interruptions

We were pleased to see such concordance between rankings independent of the particular weighting function used, and the decreasing monotonicity of rankings w.r.t. external weights of employees suggested that the model was correct and that it captured implicit relationships of perceived importance of employees. However, several criticisms were raised upon our presentation of these initial results. First and foremost, many people pointed out that the importance of a person is not always inversely proportional to the speed of reply. It is common in many workplaces that the flow of work is constantly interrupted by the influx of emails, and some of the emails, despite not being sent by important persons, are done away quickly because they require almost no time to answer. The phenomenon of having the psychological urge to "get it done" is well established in the literature, it serves the purpose of "closing"

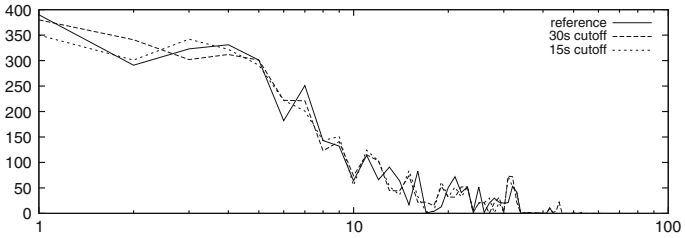


Fig. 3 Interruption cutoff ranking

open tasks and releasing the positions in the priority queue. Therefore, we have experimented with a minor tweak of the model and we have removed from employee ranking computation messages that were answered extremely quickly, marking them as interruptions. These interruptions may represent a simple acknowledgement of reading the message, a single sentence reply, a virtual handshake, etc.

Below we present the results of modifying the model by removing from consideration interruptions. We have experimented with different cutoff thresholds (30, 45, 60 s) for the interruption. Also, a problem arose with first messages in a burst: since we could not have established the time it took a given person to answer, we could either include or remove these messages.

For the clarity of presentation we have included in Fig. 3 only the best solutions. As the reference we have taken one of the ranking functions that did not employ the identification of interruptions. When we set the cutoff at 30 s (i.e., if an email took only 30 s to reply, we assume that it was not an important message, and thus it does not contribute to the perceived importance of the sender), the resulting ranking is almost strictly monotonically decreasing, which in turn means, that the ranking of employees derived from their behavioral traits is almost identical to the artificial ranking of employees based on their administrative positions. We have also found that the removal of messages for which it was impossible to compute the amount of time necessary to reply did not contribute to the final ranking. This counterintuitive result can be easily explained by the fact that this removal included around 50 % of all messages and the loss of information due to cleaning could not be compensated by more accurate importance estimations.

5.2 Effort

Our second improvement over the initial model was the inclusion of effort computation. Many critics rightly pointed out that the most pronounced feature of emails sent to important people was the effort in email preparation. Indeed, when replying to an email that one deems important (i.e., when replying to a person that one considers to be important), it is common to put more work into the contents, proper spelling and punctuation, clarity of language used, comprehensiveness, etc. Thus, we have further

extended our model by including the effort of message preparation into the calculation of edge weight. We compute the effort of message preparation based on the time between consecutive messages that occur in a *burst*. We say that two messages m_{ij}^{t} and $m_{ij}^{t'}$ occur in a burst if the time between the two messages does not extend a predefined threshold, $\left|t - t'\right| < \xi$. We have experimented with different values of the threshold ξ and we have found that the best results could be obtained with the burst of the length of 45 min, which means, that there are no email messages that take more than 45 min of continuous work. Because we cannot compute the effort of the first message in each burst, we have decided to discard this message from rank computation. With these explanations in mind we can proceed to the formal definition of the effort.

$$e(m_{ij}^{t}) = \left|t - t'\right|, \quad \left|t - t'\right| < \xi, \quad t' : \exists m_{ik}^{t'} \wedge \nexists m_{ik}^{t''} : t' < t'' < t \tag{8}$$

As can be seen in Eq. 8, the effort of the message m_{ij}^{t} is defined as the time span between sending of the message and the directly preceding message $m_{ik}^{t'}$, provided that this time span is within the limits defined by the burst.

Let us now see how the inclusion of the effort affects the resulting rankings of employees. In the figures below we have used, as the reference base weight w_r, the QN ranking with the length of the queue set at 15 (recall that the Queue Number ranking assigns the weight to the message based on the position of the message in the message queue, i.e., $w_r = \max\{0, 15 - (|Q| - q_i)\}$, where w_r denotes the reference weight of the message, and q_i denotes the position of the message at the moment of reply. We have verified the following ways of including effort into ranking computation:

5.2.1 E_1 Scaled Multiplied Effort

The weight of the message m_{ij}^{t} is $w = \lambda e(m_{ij}^{t}) w_r$, where λ denotes the scaling factor which defines the ratio between the importance of the effort w.r.t. the importance of the reference weight computed from the QN ranking. In the results below we have set $\lambda = 3$, and $e, w_r \in \langle 1, 2\rangle$. This formula tries to stress the importance of effort as compared to the ordering of replies.

5.2.2 E_2 Simple Multiplied Effort

Here the weight of the message m_{ij}^{t} is $w = e(m_{ij}^{t}) w_r$, with $e, w_r \in \langle 1, 2\rangle$. It is a simplified version of the scaled multiplication with $\lambda = 1$. We use this ranking to verify if putting more emphasis on the effort (by increasing the coefficient λ) is of any benefit.

5.2.3 E_3 Power Effort

The weight of the message m_{ij}^t is $w = (e(m_{ij}^t))^2 w_r$. We are trying to establish if putting even more emphasis on the effort by increasing its influence nonlinearly with the increase in effort would produce a better ranking. The danger with this scheme is that the influence of outliers (i.e. messages that took much time to answer due to some external reason not related to the real importance of the sender) can bias the final ranking significantly.

5.2.4 E_4 Additive Effort

The weight of the message m_{ij}^t is $w = e(m_{ij}^t) + w_r$, with $e, w_r \in \langle 0, 1 \rangle$. According to this scheme we treat the original weight obtained from the QN ranking and the computed effort equally, after normalizing both values to the common range of $\langle 0, 1 \rangle$.

5.2.5 E_5 Additive Effort with Addressee Weighting

This scheme is very similar to the previous scheme E_4. The weight of the message m_{ij}^t is $w = (e(m_{ij}^t) + w_r)R(u_j)$, where $R(u_j)$ denotes the current ranking of the user u_j who is the addressee of the message. As previously, both effort and base weight are normalized to the interval $\langle 0, 1 \rangle$. If indeed one puts more effort in replying to people generally considered important by the entire population, this weighting scheme should outperform other schemes. However, the caveat here is that $R(u_j)$ denotes the global ranking of the user u_j, which does not necessarily have to be in accordance with the way a particular user perceives the importance of the user u_j.

5.2.6 E_6 Effort Only

The last scheme defines the weight of the message m_{ij}^t simply as $w = e(m_{ij}^t)$. We discard the information on the base weight w_r and we compute the importance of the message solely based on the effort of message preparation. This approach is prone to many errors because there are multiple factors that may contribute to the time it took to answer a message that were not related to the importance of the message (e.g., a person was interrupted during email writing) and we have no way of finding this from email communication data.

Figure 4 presents the results of the above described schemes of including effort into ranking computation. We see that, in general, including the effort of the message into ranking computation is beneficial. In particular, we observe that additive effort schemes (E_4 and E_5), and to some extent the effort only scheme (E_6), provide a more monotonically decreasing rankings, while multiplied effort schemes (E_1 and E_2) and power effort scheme (E_3) do not improve over the base QN ranking.

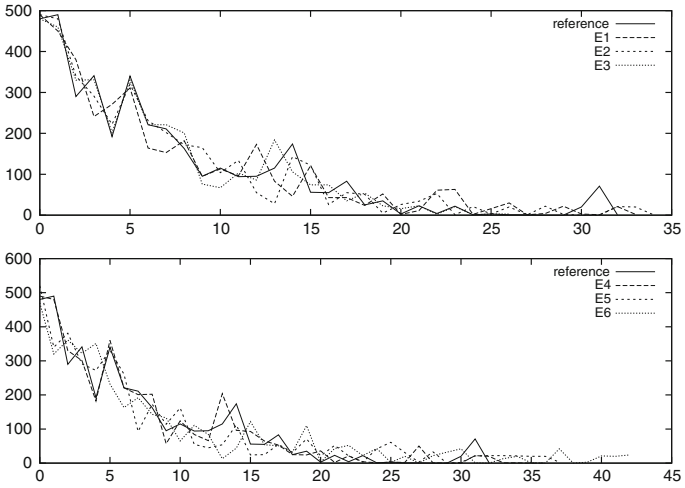


Fig. 4 Effort rankings

6 Conclusions

In this chapter we have presented a novel method for assessing the importance of actors in the social network. The resulting ranking may be used as an alternative centrality measure of actors. The method is based on the analysis of email communication patterns. In particular, we are interested in measuring the delay in answering each message. We make the assumption that actors have an implicit ranking of the importance of other actors and that this implicit ranking can be discovered by measuring the procrastination in answering emails. The experiments conducted on a large body of data representing a one year of email communication within the Institute of Computing Science of Poznan University of Technology prove the validity of the proposed solution. The centrality ranking produced by our method has an additional benefit of being fully personalized, i.e., it is possible both to aggregate individual importance rankings to obtain a single global importance ranking, but at the same time one may use individual importance rankings to order other actors of the network independently for each actor.

The results presented in this chapter are preliminary. As our future work we intend to verify other weighting schemes. We also want to gather data from other organizations and to verify if our model is universal enough to be employed in other environments. In particular, we believe that the approach consisting in measuring the effort required to perform an action to assess the individual perception of importance of the action can be successfully used in many different application domains. We are currently investigating the applicability of our method to rank events (posts, comments and likes) in the social network.

Acknowledgments This research has been supported by the National Science Centre grant 2011/03/B/ST6/01563.

References

1. Bakshy E, Hofman J, Mason W, Watts D (2011) Everyone's an influencer: quantifying influence on twitter. In: Proceedings of the fourth ACM international conference on Web search and data mining, ACM, pp 65–74
2. Bar-Ilan J (2005) Comparing rankings of search results on the web. Inf Process Manage 41(6):1511–1519
3. Bar-Ilan J, Mat-Hassan M, Levene M (2006) Methods for comparing rankings of search engine results. Comput Netw 50(10):1448–1463
4. Barabasi A (2005) The origin of bursts and heavy tails in human dynamics. Nature 435(7039):207–211
5. Barabási A, Frangos J (2002) Linked: the new science of networks science of networks. Basic Books, New York
6. Barabâsi A, Jeong H, Néda Z, Ravasz E, Schubert A, Vicsek T (2002) Evolution of the social network of scientific collaborations. Phys A: Stat Mech Appl 311(3):590–614
7. Bird C, Gourley A, Devanbu P, Gertz M, Swaminathan A (2006) Mining email social networks. In: Proceedings of 2006 international workshop on Mining software repositories, ACM, pp 137–143
8. Boyd D, Golder S, Lotan G (2010) Tweet, tweet, retweet: conversational aspects of retweeting on twitter. In: Proceedings of 2010 43rd Hawaii international conference on system sciences (HICSS), IEEE, pp 1–10
9. Cha M, Haddadi H, Benevenuto F, Gummadi K (2010) Measuring user influence in twitter: the million follower fallacy. In: Proceedings of the 4th international AAAI conference on weblogs and social media (ICWSM), vol 14. p 8
10. Culotta A, Bekkerman R, Mccallum A (2004) Extracting social networks and contact information from email and the web. In: Proceedings of CEAS-1
11. Diesner J, Frantz T, Carley K (2005) Communication networks from the enron email corpus itt's always about the people. enron is no different. Comput Math Organ Theor 11(3):201–228
12. Dwork C, Kumar R, Naor M, Sivakumar D (2001) Rank aggregation methods for the web. In: Proceedings of the 10th international conference on World Wide Web, ACM, pp 613–622
13. Fagin R, Kumar R, Sivakumar D (2003) Comparing top k lists. SIAM J Discrete Math 17(1):134–160
14. Golbeck J, Hendler J (2004) Reputation network analysis for email filtering. In: Proceedings of the first conference on email and anti-spam, vol 44. pp 54–58
15. Kim J, Masuda N, Kahng B (2009) A priority queue model of human dynamics with bursty input tasks. In: Zhou J (ed) Complex sciences, vol 5., Lecture notes of the Institute for Computer Sciences, Social Informatics and Telecommunications EngineeringSpringer, Berlin Heidelberg, pp 2402–2410
16. Kumar R, Vassilvitskii S (2010) Generalized distances between rankings. In: Proceedings of the 19th international conference on World Wide Web, ACM, pp 571–580
17. Lubarski P, Morzy M (2012) Measuring the importance of users in a social network based on email communication patterns. In: Proceedings of 2012 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM), IEEE, pp 86–90
18. Malaga R (2001) Web-based reputation management systems: problems and suggested solutions. Electron Commer Res 1(4):403–417
19. Resnick P, Kuwabara K, Zeckhauser R, Friedman E (2000) Reputation systems. Commun ACM 43(12):45–48

20. Ruohomaa S, Kutvonen L, Koutrouli E (2007) Reputation management survey. In: Availability, In: Proceedings of the second international conference on reliability and security, 2007, ARES 2007, IEEE, pp 103–111
21. Trier M, Bobrik A (2007) Analyzing the dynamics of community formation using brokering activities. In: Communities and technologies 2007. Springer, Berlin, pp 463–477
22. Weng J, Lim E, Jiang J, He Q (2010) Twitterrank: finding topic-sensitive influential twitterers. In: Proceedings of the third ACM international conference on Web search and data mining, ACM, pp 261–270

Twitter Sentiment Analysis: How to Hedge Your Bets in the Stock Markets

Tushar Rao and Saket Srivastava

Abstract Emerging interest of trading companies and hedge funds in mining social web has created new avenues for intelligent systems that make use of public opinion in driving investment decisions. It is well accepted that at high frequency trading, investors are tracking memes rising up in microblogging forums to count for the public behavior as an important feature while making short term investment decisions. We investigate the complex relationship between tweet board literature (like bullishness, volume, agreement etc) with the financial market instruments (like volatility, trading volume and stock prices). We have analyzed Twitter sentiments for more than 4 million tweets between June 2010 and July 2011 for DJIA, NASDAQ-100 and 11 other big cap technological stocks. Our results show high correlation (upto 0.88 for returns) between stock prices and twitter sentiments. Further, using Granger's Causality Analysis, we have validated that the movement of stock prices and indices are greatly affected in the short term by Twitter discussions. Finally, we have implemented Expert Model Mining System (EMMS) to demonstrate that our forecasted returns give a high value of R-square (0.952) with low Maximum Absolute Percentage Error (MaxAPE) of 1.76 % for Dow Jones Industrial Average (DJIA). We introduce and validate performance of market monitoring elements derived from public mood that can be exploited to retain a portfolio within limited risk state during typical market conditions.

T. Rao (✉)
Netaji Subhas Institute of Technology, Delhi, India
e-mail: rao.tushar@nsitonline.in

S. Srivastava
IIIT-D, Delhi, India
e-mail: saket@iiitd.ac.in

F. Can et al. (eds.), *State of the Art Applications of Social Network Analysis*, Lecture Notes in Social Networks, DOI: 10.1007/978-3-319-05912-9_11,

1 Introduction

Financial analysis and computational finance have been an active area of research for many decades [17]. Over the years, several new tools and methodologies have been developed that aim to predict the direction as well as range of financial market instruments as accurately as possible [16]. Before the emergence of internet, information regarding company's stock price, direction and general sentiments took a long time to disseminate among people. Also, the companies and markets took a long time (weeks or months) to calm market rumors, news or false information (memes in Twitter context). Web 3.0 is characterized with fast pace information dissemination as well as retrieval [6]. Spreading good or bad information regarding a particular company, product, person etc. can be done at the click of a mouse [1, 7] or even using micro-blogging services such as Twitter [4]. Recently scholars have made use of twitter feeds in predicting box office revenues [2], political game wagons [29], rate of flu spread [27] and disaster news spread [11]. For short term trading decisions, *short term sentiments* play a very important role in *short term performance* of financial market instruments such as indexes, stocks and bonds [24].

Early works on stock market prediction can be summarized to answer the question—Can stock prices be really predicted? There are two theories—(1) random walk theory (2) and efficient market hypothesis (EMH) [22]. According to EMH stock index largely reflect the already existing news in the investor community rather than present and past prices. On the other hand, random walk theory argues that the prediction can never be accurate since the time instance of news is unpredictable. A research conducted by Qian et al. compared and summarized several theories that challenge the basics of EMH as well as the random walk model completely [23]. Based on these theories, it has been proven that some level of prediction is possible based on various economic and commercial indicators. The widely accepted semi-strong version of the EMH claims that prices aggregate all publicly available information and instantly reflect new public version [19]. It is well accepted that *news drive macro-economic movement in the markets*, while researches suggests that *social media buzz is highly influential at micro-economic level*, specially in the big indices like DJIA [5, 14, 20, 26]. Through earlier researches it has been validated that market is completely driven by sentiments and bullishness of the investor's decisions [23]. Thus a comprehensive model that could incorporate these sentiments as a parameter is bound to give superior prediction at *micro-economic* level.

Earlier work done by Bollen et al. shows how collective mood on Twitter (aggregate of all positive and negative tweets) is reflected in the DJIA index movements [5] and [20]. In this work we have applied simplistic message board approach by defining bullishness and agreement terminologies derived from positive and negative vector ends of public sentiment w.r.t. each market security or index terms (such as returns, trading volume and volatility) [25]. Proposed method is not only scalable but also gives more accurate measure of large scale investor sentiment that can be potentially used for short term hedging strategies as discussed ahead in Sect. 6. This gives clear distinctive way for modeling sentiments for service based companies such as Google in contrast to product based companies such as Ebay, Amazon and Netflix.

We validate that Twitter feed for any company reflects the public mood dynamics comprising of breaking news and discussions, which is causative in nature. Therefore it adversely affects any investment related decisions which are not limited to stock discussions or profile of mood states of entire Twitter feed.

In Sect. 2, we discuss the motivation of this work and related work in the area of stock market prediction in Sect. 3. In Sect. 4 we explain the techniques used in mining data and explain the terminologies used in market and tweet board literature. In Sect. 5 we have given prediction methods used in this model with the forecasting results. In Sect. 6 we discuss how Twitter based model can be used for improving hedging decisions in a diversified portfolio by any trader. Finally in Sect. 7 we discuss the results and in Sect. 8 we present the future prospects and conclude the work.

2 Motivation

> Communities of active investors and day traders who are sharing opinions and in some case sophisticated research about stocks, bonds and other financial instruments will actually have the power to move share prices ...making Twitter-based input as important as any other data related to the stock
>
> –TIME (2009) [21]

High Frequency Trading (HFT) comprises of very high percentage of trading volumes in the present US stock exhange. Traders make an investment position that is held only for very brief periods of time—sometimes only for a few seconds. Investors rapidly trades into and out of those positions, sometimes thousands or tens of thousands of times a day. Therefore the value of an investment is as good as the last known index price. Most investors will make use of anything that will give them an advantage in placing market bets. A large percentage of high frequency traders have trained AI bots to capture buzzing trends in the social media feeds *without learning dynamics of the sentiment and accurate context of the deeper information being diffused in the social networks.* For example, in February 2011 during Oscars when Anne Hathaway was trending, stock prices of Berkshire Hathaway rose by 2.94 % [28]. Figure 1 highlight the incidents when the stock price of Berkshire Hathaway jumped coinciding with an increase of buzz on social networks/ micro-blogging websites regarding Anne Hathaway (for example during movie releases).

The events are marked as red points in the Fig. 1, event specific news on the points:
A: Oct. 3, 2008—Rachel Getting Married opens: BRK.A up 0.44 %
B: Jan. 5, 2009 — Bride Wars opens: BRK.A up 2.61 %
C: Feb. 8, 2010—Valentine's Day opens: BRK.A up 1.01 %
D: March 5, 2010—Alice in Wonderland opens: BRK.A up 0.74 %
E: Nov. 24, 2010—Love and Other Drugs opens: BRK.A up 1.62 %
F: Nov. 29, 2010—Anne announced as co-host of the Oscars: BRK.A up 0.25 %
G: Feb. 28. 2011—Anne hosts Oscars with James Franco: BRK.A up 2.94 %

Fig. 1 Historical chart of Berkshire Hathaway (BRK.A) stock over the last 3 years. Highlighted points (*A*–*F*) are the days when its stock price jumped due to an increased news volume on social networks and Twitter regarding Anne Hathaway. Courtesy Google Finance

As seen in this example, large volume of tweets can create *short term influential effects* on stock prices. Simple observations such as these motivate us to investigate deeper relationship between the dynamics of social media messages and market movements [17]. This work is not directed towards finding a new stock prediction technique, which would certainly include effects of various other macroeconomic factors. *The aim of this work*, is to quantitatively evaluate the *effects of twitter sentiment dynamics* around a stocks indices/stock prices and use it in conjunction with the *standard* model to improve the accuracy of prediction. Further in Sect. 6 we investigate into how tweets can be very useful in identifying trends in futures and options markets and to build hedging strategies to protect one's investment position in the shorter term.

3 Related Work

There have been several works related to web mining of data (blogposts, discussion boards and news) [3, 12, 14] and to validate the significance of assessing behavioral changes in the public mood to track movements in stock markets. Some trivial work shows information from investor communities is causative of speculation regarding private and forthcoming information and commentaries [8, 9, 18, 30]. Dewally in 2003 worked upon naive momentum strategy confirming recommended stocks through user ratings had significant prior performance in returns [10]. But now with the pragmatic shift in the online habits of communities around the worlds, platforms like StockTwits[1] [31] and HedgeChatter[2] have come. Das and Chen made the initial attempts by using natural language processing algorithms classifying stock messages based on human trained samples. However their result did not carried statistically significant predictive relationships [9].

[1] http://stocktwits.com/

[2] http://www.hedgechatter.com/

Gilbert et al. and Zhang et al. have used corpus from livejournal blogposts in assessing the bloggers sentiment in dimensions of fear , anxiety and worry making use of Monte Carlo simulation to reflect market movements in S&P 500 index [14, 32]. Similar and significantly accurate work is done by Bollen et al. [5] who used dimensions of Google—Profile of Mood States to reflect changes in closing price of DJIA. Sprengers et al. analyzed individual stocks for S&P 100 companies and tried correlating tweet features about discussions of the stock discussions about the particular companies containing the Ticker symbol [26]. However these approaches have been restricted to community sentiment at macro-economic level which doesn't give explanatory dynamic system for individual stock index for companies as discussed in our previous work [25]. Thus deriving a model that is scalable for individual stocks/companies and can be exploited to make successful hedging strategies as discussed in Sect. 6.

4 Web Mining and Data Processing

In this section we describe our method of Twitter and financial data collection as shown in Fig. 2. In the first phase, we mine the tweet data and after removal of spam/noisy tweets, they are subsequently subjected to sentiment assessment tools in phase two. In later phases feature extraction, aggregation and analysis is done.

4.1 Tweets Collection and Processing

Out of other investor forums and discussion boards, Twitter has widest acceptance in the financial community and all the messages are accessible through a simple search of requisite terms through an application programming interface (API)[3]. Sub forums of Twitter like StockTwits and TweetTrader have emerged recently as hottest place for investor discussion buy/sell out at voluminous rate. Efficient mining of sentiment aggregated around these tweet feeds provides us an opportunity to trace out relationships happening around these market sentiment terminologies. Currently more than 250 million messages are posted on Twitter everyday (Techcrunch October 2011[4]).

This study was conducted over a period of 14 months period between June 2nd 2010 to 29th July 2011. During this period, we collected 4,025,595 (by around 1.08 M users) English language tweets Each tweet record contains tweet identifier, date/time of submission (in GMT), language and text. Subsequently the stop words

[3] Twitter API is easily accessible through an easy documentation available at—https://dev.twitter.com/docs Also Gnip—http://gnip.com/twitter, the premium platform available for purchasing public firehose of tweets has many investors as financial customers researching in the area.

[4] http://techcrunch.com/2011/10/17/twitter-is-at-250-million-tweets-per-day/

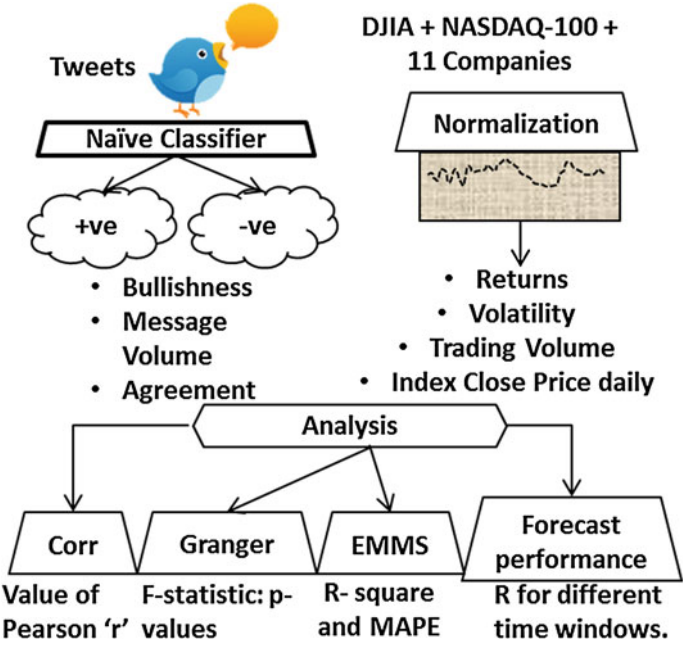


Fig. 2 Flowchart of the proposed methodology. Four set of results obtained (1) correlation results for twitter sentiments and stock prices for different companies (2) Granger's casuality analysis to causation (3) Using EMMS for quantitative comparison (4) Performance of forecasting method over different time windows

Table 1 List of companies

Company name	Ticker symbol
Amazon	AMZN
Apple	AAPL
AT&T	T
Dell	DELL
EBay	EBAY
Google	GOOG
Microsoft	MSFT
Oracle	ORCL
Samsung Electronics	SSNLF
SAP	SAP
Yahoo	YHOO

and punctuation are removed and the tweets are grouped for each day (which is the highest time precision window in this study since we do not group tweets further based on hours/minutes). We have directed our focus DJIA, NASDAQ-100 and 11 major companies listed in Table 1. These companies are some of the highly traded and discussed technology stocks having very high tweet volumes.

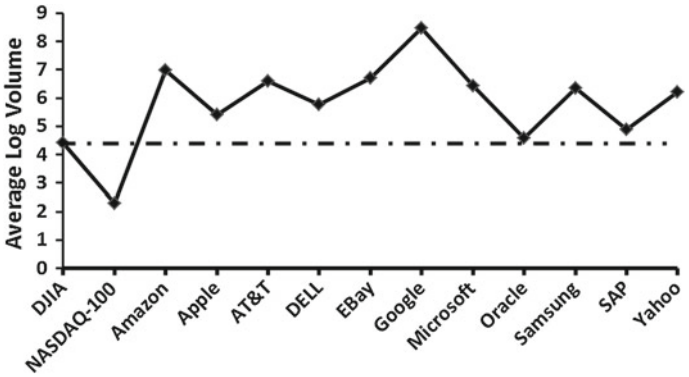


Fig. 3 Graph for average of log of daily volume over the months under study

11 tech companies are selected on the basis of average message volume. If their average tweet volume is more than the tweet discussion volume of DJIA and NASDAQ-100, they are included in the analysis, as observed in the Fig. 3. In this study we have observed that technology stocks generally have a higher tweet volume than non-technology stocks. One reason for this may be that all technology companies come out with new products and announcements much more frequently than companies in other sectors (say infrastructure, energy, FMCG, etc.) thereby generating greater buzz on social media networks. However, our model may be applied to any company/indices that generate high tweet volume.

4.2 Sentiment Classification

In order to compute sentiment for any tweet we had to classify each incoming tweet everyday into *positive* or *negative* using nave classifier. For each day total number of positive tweets is aggregated as $Positive_{day}$ while total number of negative tweets as $Negative_{day}$. We have made use of JSON API from Twittersentiment,[5] a service provided by Stanford NLP research group [15]. Online classifier has made use of Naive Bayesian classification method, which is one of the successful and highly researched algorithms for classification giving superior performance to other methods in context of tweets. Their classification training was done over a dataset of 1,600,000 tweets and achieved an accuracy of about 82.7 %. These methods have high replicability and few arbitrary fine tuning elements.

In our dataset roughly 61.68 % of the tweets are positive, while 38.32 % of the tweets are negative for the company stocks under study. The ratio of 3:2 indicates stock discussions to be much more balanced in terms of bullishness than internet board messages where the ratio of positive to negative ranges from 7:1 [10] to 5:1 [12].

[5] https://sites.google.com/site/twittersentimenthelp/

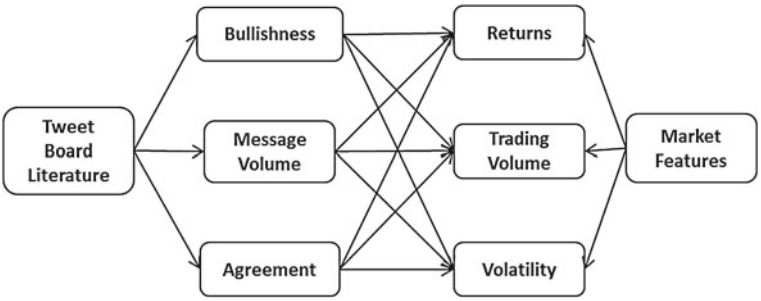


Fig. 4 Tweet sentiment and market features

Balanced distribution of stock discussion provides us with more confidence to study information content of the positive and negative dimensions of discussion about the stock prices on microblogs.

4.3 Tweet Feature Extraction

One of the research questions this study explores is how investment decisions for technological stocks are affected by entropy of information spread about companies under study in the virtual space. Tweet messages are micro-economic factors that affect stock prices which is quite different type of relationship than factors like news aggregates from traditional media, chatboard room etc. which are covered in earlier studies over a particular period [10, 12, 18]. Keeping this in mind we have only aggregated the tweet parameters (extracted from tweet features) over a day. In order to calculate parameters weekly, bi-weekly, tri-weekly, monthly, 5 weekly and 6 weekly we have simply taken average of daily twitter feeds over the requisite period of time.

Twitter literature in perspective of stock investment is summarized in Fig. 4. We have carried forward work of Antweiler et al. for defining bullishness (B_t) for each day (or time window) given equation as:

$$B_t = \ln\left(\frac{1 + {M_t}^{Positive}}{1 + {M_t}^{Negative}}\right) \tag{1}$$

Where ${M_t}^{Positive}$ and ${M_t}^{Negative}$ represent number of positive or negative tweets on a particular day t. Logarithm of bullishness measures the share of surplus positive signals and also gives more weight to larger number of messages in a specific sentiment (positive or negative). Message volume for a time interval t is simply defined as natural logarithm of total number of tweets for a specific stock/index which is $\ln({M_t}^{Positive} + {M_t}^{Negative})$. The agreement among positive and negative tweet messages is given by:

$$A_t = 1 - \sqrt{(1 - | \frac{(M_t^{Positive} - M_t^{Negative})}{(M_t^{Positive} + M_t^{Negative})} |} \quad (2)$$

If *all* tweet messages about a particular company are bullish or bearish, agreement would be 1 in that case. Influence of silent tweets days in our study (trading days when no tweeting happens about particular company) is less than 0.1 % which is significantly less than previous research [12, 26]. Carried terminologies for all the tweet features {Positive, Negative, Bullishness, Message Volume, Agreement} remain same for each day with the lag of one day. For example, carried bullishness for day d is given by $CarriedBullishness_{d-1}$.

4.4 Financial Data Collection

We have downloaded financial stock prices at daily intervals from Yahoo Finance API[6] for DJIA, NASDAQ-100 and the companies under study given in Table 1. The financial features (parameters) under study are opening (O_t) and closing (C_t) value of the stock/index, highest (H_t) and lowest (L_t) value of the stock/index and returns. Returns are calculated as the difference of logarithm to the base e between the closing values of the stock price of a particular day and the previous day.

$$R_t = \{\ln Close_{(t)} - \ln Close_{(t-1)}\} \times 100 \quad (3)$$

Trading volume is the logarithm of number of traded shares. We estimate daily volatility based on intra-day highs and lows using Garman and Klass volatility measures [13] given by the formula:

$$\sigma = \sqrt{\frac{1}{n}\sum \frac{1}{2}\left[\ln \frac{H_t}{L_t}\right]^2 - [2\ln 2 - 1]\left[\ln \frac{C_t}{O_t}\right]^2} \quad (4)$$

5 Statistical Analysis and Results

We begin our study by identifying the correlation between the Twitter feed features and stock/index parameters which give the encouraging values of statistically significant relationships with respect to individual stocks(indices). To validate the causative effect of tweet feeds on stock movements we have used econometric technique of Granger's Casuality Analysis. Furthermore, we make use of expert model mining system (EMMS) to propose an efficient prediction model for closing price of DJIA and NASDAQ 100. Since this model does not allow us to draw conclusion

[6] http://finance.yahoo.com/

about the accuracy of prediction (which will differ across size of the time window) subsequently discussed later in this section.

5.1 Correlation Matrix

For the stock indices DJIA and NASDAQ and 11 tech companies under study we have come up with the correlation matrix as heatmap given in Fig. 5 between the financial market and Twitter sentiment features explained in Sect. 4. Financial features for each stock/index (Open, Close, Return, Trade Volume and Volatility) is correlated with Twitter features (Positive, Negative, Bullishness, Carried Positive, Carried Negative and Carried Bullishness). The time period under study is monthly average as it the most accurate time window that gives significant values as compared to other time windows which is discussed later Sect. 5.4. Heatmap in Fig. 5 indicative of significant relationships between various twitter features with the index features.

Our approach shows strong correlation values between various features (upto -0.96 for opening price of Oracle and 0.88 for returns from DJIA index etc.) and the average value of correlation between various features is around 0.5. Comparatively highest correlation values from earlier work has been around 0.41 [26]. As the relationships between the stock(index) parameters and Twitter features show different behavior in magnitude and sign for different stocks(indices), a uniform standardized model would not applicable to all the stocks(indices). Therefore, building an individual model for each stock(index) is the correct approach for finding appreciable insight into the prediction techniques. Trading volume is mostly governed by agreement values of tweet feeds as -0.7 for same day agreement and -0.65 for DJIA. Returns are mostly correlated to same day bullishness by 0.61 and by lesser magnitude 0.6 for the carried bullishness for DJIA. Volatility is again dependent on most of the Twitter features, as high as 0.77 for same day message volume for NASDAQ-100.

One of the *anomalies* that we have observed is that EBay gives negative correlation between the all the features due to heavy product based marketing on Twitter which turns out as not a correct indicator of average growth returns of the company itself.

5.2 Bivariate Granger Causality Analysis

The results in previous section show strong correlation between financial market parameters and Twitter sentiments. However, the results also raise a point of discussion: Whether market movements affects Twitter sentiments or Twitter features causes changes in the markets? We make use of Granger Causality Analysis (GCA) to the time series averaged to weekly time window to returns through DJIA and NASDAQ-100 with the Twitter features (positive, negative, bullishness, message volume and agreement). Granger Causality Analysis (GCA) is not used to establish causality, but as an economist tool to investigate a statistical pattern of lagged correlation. A similar observation that the clouds precede rain is widely accepted. GCA

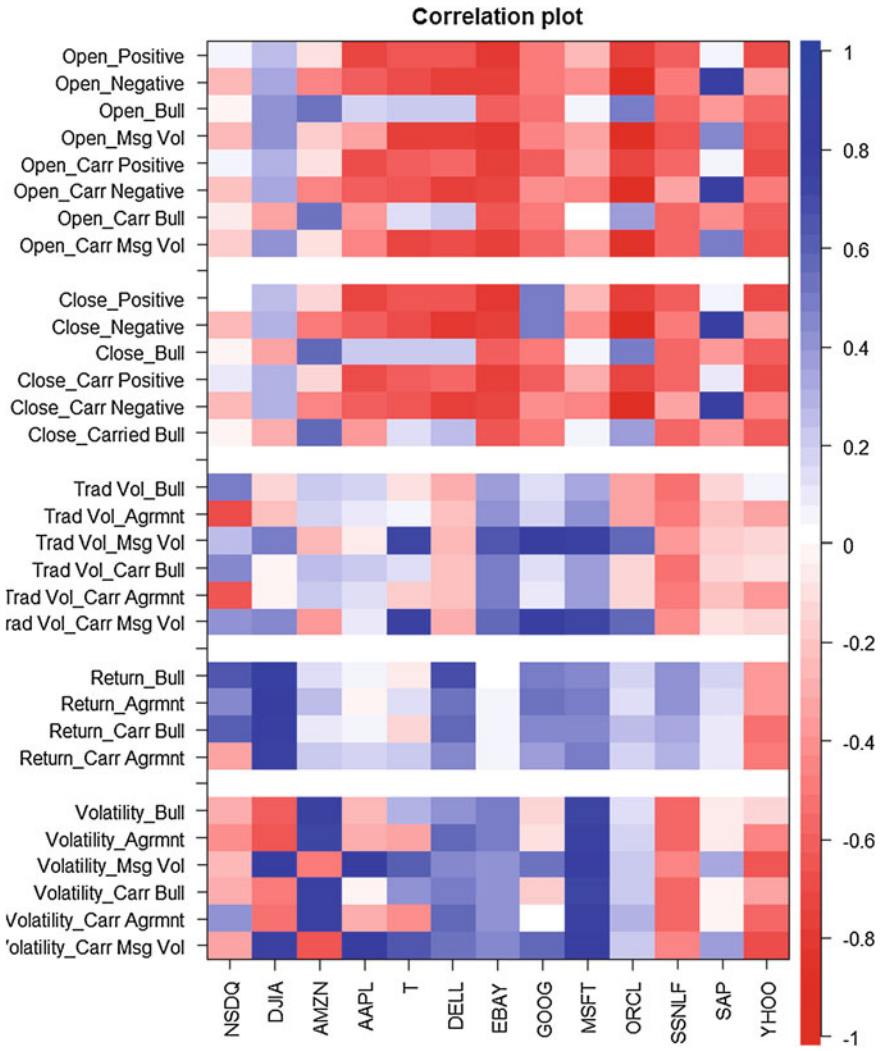


Fig. 5 Heatmap showing Pearson correlation coefficients between security indices versus features from Twitter

rests on the assumption that if a variable X causes Y then changes in X will be systematically occur before the changes in Y. We realize lagged values of X shall bear significant correlation with Y. However correlation is not necessarily behind causation. We have made use of GCA in similar fashion as [5, 14] This is to test if one time series is significant in predicting another time series. Let returns R_t be reflective of fast movements in the stock market. To verify the change in returns with the change in Twitter features we compare the variance given by following linear models:

$$R_t = \alpha + \sum_{i=1}^{n} \beta_i D_{t-i} + \varepsilon_t \tag{5}$$

$$R_t = \alpha + \sum_{i=1}^{n} \beta_i D_{t-i} + \sum_{i=1}^{n} \gamma_i X_{t-i} + \varepsilon_t \tag{6}$$

Equation 5 uses only 'n' lagged values of R_t, i.e. (R_{t-1}, ..., R_{t-n}) for prediction, while Eq. 6 uses the n lagged values of both R_t and the tweet features time series given by X_{t-1}, ..., X_{t-n}. We have taken weekly time window to validate the casuality performance, hence the lag values[7] will be calculated over the weekly intervals 1, 2, ..., 7. From the Table 2, we can reject the null hypothesis (H_o) that *the Twitter features do not affect returns in the financial markets* i.e. $\beta_{1,2,...,n} \neq 0$ with a high level of confidence (P-alues closer to zero signify stronger causative relationship). However as we see the result applies to only specific negative and positive tweets. Other features like agreement and message volume do not have significant casual relationship with the returns of a stock index (high p-values).

5.3 EMMS Model for Forecasting

We have used Expert Model Mining System (EMMS) which incorporates a set of competing methods such as Exponential Smoothing (ES), Auto Regressive Integrated Moving Average (ARIMA) and seasonal ARIMA models. In this work, selection criterion for the EMMS is coefficient of determination (R squared) which is square of the value of pearson-'r' of fit values (from the EMMS model) and actual observed values. Mean absolute percentage error (MAPE) and maximum absolute percentage error (MaxPAE) are mean and maximum values of error (difference between fit value and observed value in percentage). To show the performance of tweet features in prediction model, we have applied the EMMS twice—first with tweets features as independent predictor events and second time without them. This provides us with a quantitative comparison of improvement in the prediction using tweet features.

In the dataset we have time series for a total of approximately 60 weeks (422 days), out of which we use approximately 75 % i.e. 45 weeks for the training both the models with and without the predictors for the time period June 2nd 2010 to April 14th 2011. Further we verify the model performance as one step ahead forecast over the testing period of 15 weeks from April 15th to 29th July 2011 which count for wide and robust range of market conditions. Forecasting accuracy in the testing period is compared for both the models in each case in terms of maximum absolute percentage error (MaxAPE), mean absolute percentage error (MAPE) and the direction accuracy. MAPE is given by the Eq. 7, where $\hat{y_i}$ is the predicted value and y_i is the actual value.

[7] *lag at k* for any parameter M at x_t week is the value of the parameter prior to x_{t-k} week. For example, value of returns for the month of April, at the lag of one month will be $return_{april-1}$ which will be $return_{march}$.

Table 2 Granger's casuality analysis of DJIA and NASDAQ for 7 week lag Twitter sentiments

Index ⇓	Lag	Positive	Negative	Bull	Agrmnt	Msg vol	Carr positive	Carr negative	Carr bull	Carr agrmnt	Carr msg vol
DJIA	1	0.614	0.122	0.891	0.316	0.765	0.69	0.103	0.785	0.759	0.934
	2	0.033**	0.307	0.037**	0.094*	0.086**	0.032**	0.301**	0.047**	0.265	0.045**
	3	0.219	0.909	0.718	0.508	0.237	0.016**	0.845	0.635	0.357	0.219
	4	0.353	0.551	0.657	0.743	0.743	0.116	0.221	0.357	0.999	0.272
	5	0.732	0.066	0.651	0.553	0.562	0.334	0.045**	0.394	0.987	0.607
	6	0.825	0.705	0.928	0.554	0.732	0.961	0.432	0.764	0.261	0.832
	7	0.759	0.581	0.809	0.687	0.807	0.867	0.631	0.987	0.865	0.969
NSDQ	1	0.106	0.12	0.044**	0.827	0.064*	0.02**	0.04**	0.043**	0.704	0.071*
	2	0.048**	0.219	0.893	0.642	0.022**	0.001**	0.108	0.828	0.255	0.001**
	3	0.06*	0.685	0.367	0.357	0.135	0.01**	0.123	0.401	0.008**	0.131
	4	0.104	0.545	0.572	0.764	0.092*	0.194	0.778	0.649	0.464	0.343
	5	0.413	0.997	0.645	0.861	0.18	0.157	0.762	0.485	0.945	0.028
	6	0.587	0.321	0.421	0.954	0.613	0.795	0.512	0.898	0.834	0.591
	7	0.119	0.645	0.089	0.551	0.096	0.382	0.788	0.196	0.648	0.544

(NSDQ is short for NASDAQ. p-value < 0.01:***, p- alue < 0.05:**, p - value < 0.1:*)

Table 3 EMMS model fit characteristics for DJIA and NASDAQ-100

Index	Predictors	Model fit statistics			Ljung-Box Q(18)		
		R-squared	MaxAPE	Direction	Statistics	DF	Sig.
Dow-30	Yes	0.95	1.76	90.8	11.36	18	0.88
	No	0.92	2.37	60	9.9	18	0.94
NSDQ-100	Yes	0.68	2.69	82.8	23.33	18	0.18
	No	0.65	2.94	55.8	16.93	17	0.46

$$MAPE = \frac{\Sigma^n{}_i |\frac{y_i - \hat{y_i}}{y_i}|}{n} \times 100 \tag{7}$$

While direction accuracy is measure of how accurately market or commodity up/down movement is predicted by the model, which is technically defined as logical values for $(y_{i,\hat{t}+1} - y_{i,t}) \times (y_{i,t+1} - y_{i,t}) > 0$ respectively.

As we can see in the Table 3, there is significant reduction in MaxAPE for DJIA(2.37 to 1.76) and NASDAQ-100 (2.96 to 2.69) when EMMS model is used with predictors as events which in our case our all the Tweet features (positive, negative, bullishness, message volume and agreement). There is significant decrease in the value of MAPE for DJIA which is 0.8 in our case than 1.79 for earlier approaches [5]. As we can from the values of R-square, MAPE and MaxAPE in Table 3 for both DJIA and NASDAQ 100, our proposed model uses Twitter sentiment analysis for a superior performance over traditional methods. Figure 6 shows the EMMS model fit for weekly closing values for DJIA and NASDAQ 100. In the figure fit are model fit values, observed are values of actual index and UCL & LCL are upper and lower confidence limits of the prediction model.

5.4 Prediction Accuracy Using OLS Regression

Our results in the previous section showed that forecasting performance of stocks/indices using Twitter sentiments varies for different time windows. Hence it is important to quantitatively deduce a suitable time window that will give us most accurate prediction. Figure 7 shows the plot of R-square metric for OLS regression for returns from stock indexes NASDAQ-100 and DJIA from tweet board features (like number of positive, negative, bullishness, agreement and message volume) both for carried (at 1-day lag) and same week. From the Fig. 7 it can be inferred as we increase the time window the accuracy in prediction increases but only till a certain point that is monthly in our case beyond which value of R-square starts decreasing again. *Thus, for monthly predictions we have highest accuracy in predicting anomalies in the returns from the tweet board features*.

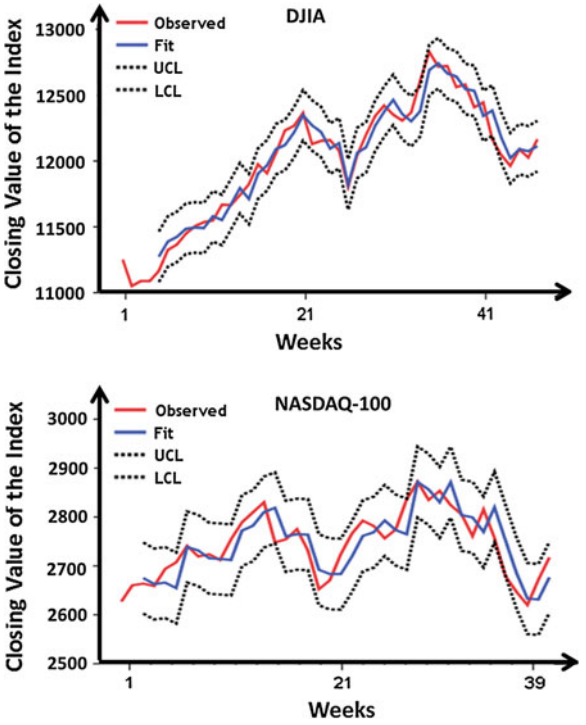


Fig. 6 Plot of Fit values (from the EMMS model) and actual observed closing values for DJIA and NASDAQ-100

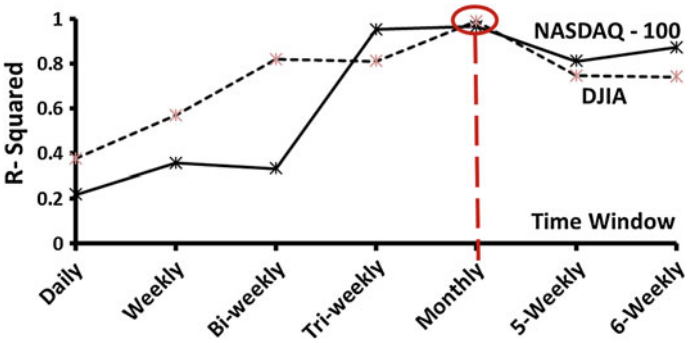


Fig. 7 Plot of R-square values over different time windows for DJIA and NASDAQ-100. Higher values denote greater prediction accuracy

6 Hedging Strategy Using Twitter Sentiment Analysis

Portfolio protection is very important practice that is weighted as much as portfolio appreciation. Just like a normal user purchases insurance for its house, car or any commodity, one can also buy insurance for the investment that is made in the stock securities. This doesn't prevent a negative event from happening, but if it does happen and you're properly hedged, the impact of the event is reduced. In a diverse portfolio hedging against investment risk means strategically using instruments in the market to offset the risk of any adverse price movements. Technically, to hedge investor invests in two securities with negative correlations, which again in itself is time varying dynamic statistics.

To explain how weekly forecast based on mass tweet sentiment features can be potentially useful for a singular investor, we will take help of a simple example.

Let us assume that the share for a company C1 is available for $X per share and the cost of premium for a stock option of company C1 (with strike price $X) is $Y.

A = total amount invested in shares of a company *C1* which is number of shares (let it be N) × $X

B = total amount invested in put option of company *C1* (relevant blocksize × $Y)

And always for an effective investment (N × $X) > (Blocksize × $Y)

An investor shall choose the value of N as per as their risk appetitive i.e. ratio of A:B = 2:1 (assumed in our example, will vary from from investor to investor). Which means in the rising market conditions, he would like to keep 50 % of his investment to be completely guarded, while rest 50 % are risky components; whereas in the bearish market condition he would like to keep his complete investment fully hedged by buying put options equivalent of all the investment he has made in shares for the same security. From Fig. 8, we infer for the P/L curves consisting of shares and 2 different put options for the company *C1* purchased as different time intervals [8]; hence the different premium price even with the same strike price of $X. Using married put strategy makes the investment risk free but reduces the rate of return in contrast to the case which comprises of only equity security which is completely free-fall to the market risk. Hence the success of married put strategy depends greatly on the accuracy of predicting whether the markets will rise of fall. Our proposed Tweet sentiment analysis can be highly effective in this prediction to determine accurate instances when the investor should readjust his portfolio before the actual changes happen in the market. Our proposed approach provides an innovative technique of using dynamic Twitter sentiment analysis to *exploit the collective wisdom of the crowd for minimising the risk in a hedged portfolio*. Below we summarize two different portfolio states at different market conditions (Table 4).

To check the effectiveness of our proposed tweet based hedging strategy, we run simulations and make portfolio adjustments in various market conditions (bullish,

[8] The reason behind purchase of long put options at different time intervals is because in a fully hedged portfolio, profit arrow has lower slope as compared to partially hedged portfolio (refer P/L graph). Thus the trade off between risk and security has to be carefully played keeping in mind the precise market conditions.

Table 4 Example investment breakdown in the two cases

Partially Hedged Portfolio at 50 % risk
1000 shares at price of \$X = 1,000 X
1 Block size of 500 shares put options purchased at strike price of \$X with premium of \$Y each = 500Y
Total = 1,00 X + 500 Y
Fully Hedged Portfolio at minimized risk
1000 shares at price of \$X = 1,000 X
2 Block size of 500 shares each put options purchased at strike price of \$X with premium of \$Y each = 2 × 500 Y = 1000 Y
Total = 1,000 X + 1,000 Y

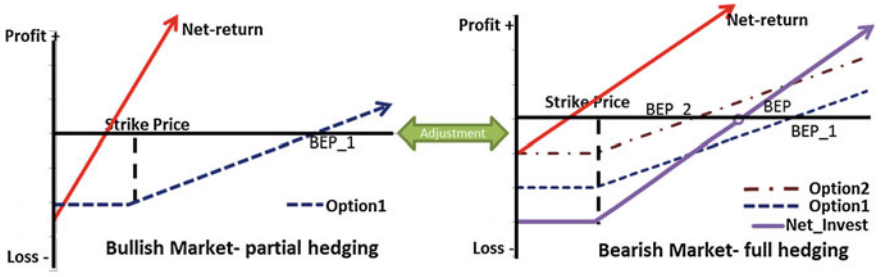


Fig. 8 Portfolio adjustment in cases of bearish (*fully hedged*) and bullish (*partial hedged*) market scenarios. In both the figures, strike price is the price at which a option is purchased, Break even point (BEP) is the instance when investment starts making profit. In case of bearish market scenario, two options at same strike price (but different premiums) are in purchased at different instances, Option1 brought at the time of initial investment and Option2 brought at a later stage (hence lower in premium value)

bearish, volatile etc). To elaborate, we take an example of DJIA ETF's as the underlying security over the time period of 14th November 2010 to 30th June 2011. Approximately 76 % of the time period is taken in the training phase to tune the SVM classifier (using tweet sentiment features from the prior week). This trained SVM classifier is then used to predict market direction (DJIA's index movement) in the coming week. Testing phase for the classification model (class 1—bullish market ↑ and class 0- bearish market ↓) is from 8th May to 30th June 2011 consisting a total of 8 weeks. SVM model is build using KSVM classification technique with the linear (vanilladot—best fit) kernel using the package 'e1071' in R statistical framework. Over the training dataset, the tuned value of the objective function is obtained as −4.24 and the number of support vectors is 8. Confusion matrix for the predicted over the actual values (in percentage) is given in Table 5. (Percentages do not sum to full 100 % as the remaining 12.5 % falls under the third type of class when the value of the index do not change. This class is excluded in the current analysis due to limitations of data period) Overall classifier accuracy over the testing phase is 85.7 %. Receiver operator characteristics (ROC) curve measuring the accuracy of

Table 5 Prediction accuracy over the testing phase (8 weeks). Values in percentage

Confusion matrix		Predicted direction	
		Market down	Market up
Actual direction	Market down	37.5	12.5
	Market up	0	37.5

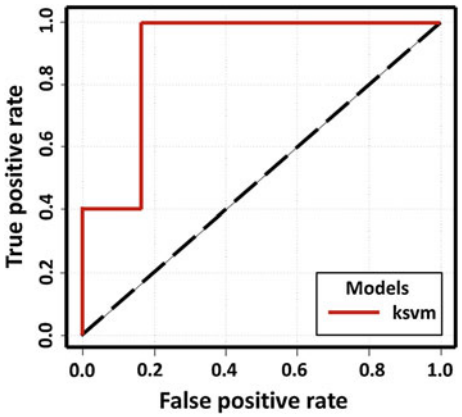


Fig. 9 Receiver operating characteristic (*ROC curve*) curve for the KSVM classifier prediction over the testing phase. ROC is graphical plot of the sensitivity or true positive rate, versus false positive rate (one minus the specificity or true negative rate). More the area under curve for typical ROC, more is the performance efficiency of the machine learning algorithm

the classifier as true positive rate to false positive rate is given in the Fig. 9. It shows the tradeoff between sensitivity i.e. true positive rate and specificity i.e. true negative rate (any increase in sensitivity will be accompanied by a decrease in specificity). Good statistical significance for the classification accuracy can be inferred from the value of area under the ROC curve (AUC) which comes out to 0.88.

Figure 10 shows the DJIA index during the testing period and the arrows mark the weeks when the adjustment is done in the portfolio based on prediction obtained from tweet sentiment analysis of prior week. At the end of the week (on Sunday), using tweet sentiment feature we predict what shall be the market condition in the coming week- whether the prices will go down or up. Based on the prediction portfolio adjustment—bearish $\longrightarrow$ bullish or bullish $\longrightarrow$ bearish.

7 Discussions

In Sect. 5, we observed how the statistical behavior of market through Twitter sentiment analysis provides dynamic window to the investor behavior. Furthermore, in the Sect. 6 we discussed how behavioral finance can be exploited in portfolio decisions to make highly reduced risked investment. Our work answers the important question—If someone is talking bad/good about a company (say Apple etc.) as singular sentiment irrespective of the overall market movement, is it going to adversely affect the stock price? Among the 5 observed Twitter message features both at same day and lagged intervals we realize only some are Granger causative of the returns

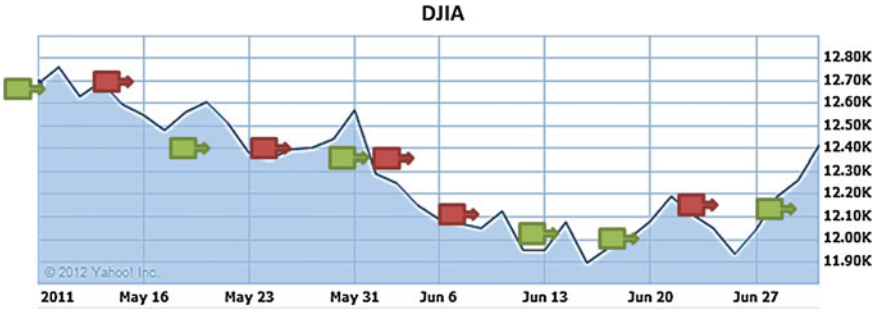


Fig. 10 DJIA index during the testing period. In the figure *green marker* shows adjustment bearish ⟶ bullish, while *red arrow* shows adjustment bullish ⟶ bearish. (Data-point at the centre of the box) (Data courtesy Yahoo! finance)

Table 6 Comparison of various approaches for modeling markets movements through Twitter

Previous approaches →	Bollen et al. [5] and Gilbert et al. [14]	Sprenger et al. [26]	This work
Approach	Mood of complete Twitter feed	Stock discussion with ticker $ on Twitter	Discussion based tracking of Twitter sentiments
Dataset	28th Feb 2008 to 19th Dec 2008, 9M tweets sampled as 1.5 % of Twitter feed	1st Jan 2010 to 30th June 2010—0.24M tweets	2nd June 2010 to 29th July 2011-4M tweets through search API
			* High corr values (upto −0.96) for opening price
			* Strong corr values (upto 0.88) for returns
Results	* 86.7 % directional accuracy for DJIA	* Max corr value of 0.41 for returns of S&P 100 stocks	* MaxAPE of 1.76% for DJIA
			* Directional accuracy of 90.8 % for DJIA
Feedback/ Drawbacks	Individual modeling for stocks not feasible	News not taken into account, very less tweet volumes	Comprehensive and customizable approach. Can be used for hedging in F&O markets

from DJIA and NASDAQ-100 indexes, while changes in the public sentiment is well reflected in the return series occurring at even lags of 1, 2 and 3 weeks. Remarkably the most significant result is obtained for returns at lag 2 (which can be inferred as possible direction for the stock/index movements in the next week).

Table 6 given below explains the different approaches to the problem that have been done in past by researchers [5, 14, 26]. As can be seen from the table, our

approach is scalable, customizable and verified over a large data set and time period as compared to other approaches. Our results are significantly better than the previous work. Furthermore, this model can be of effective use in formulating short-term hedging strategies (using our proposed Twitter based prediction model).

8 Conclusion

In this chapter, we have worked upon identifying relationships between Twitter based sentiment analysis of a particular company/index and its short-term market performance using large scale collection of tweet data. Our results show that negative and positive dimensions of public mood carry strong cause-effect relationship with price movements of individual stocks/indices. We have also investigated various other features like how previous week sentiment features control the next week's opening, closing value of stock indexes for various tech companies and major index like DJIA and NASDAQ-100. Table 6 shows as compared to earlier approaches in the area which have been limited to wholesome public mood and stock ticker constricted discussions, we verify strong performance of our alternate model that captures mass public sentiment towards a particular index or company in scalable fashion and hence empower a singular investor to ideate coherent relative comparisons. Our analysis of individual company stocks gave strong correlation values (upto 0.88 for returns) with twitter sentiment features of that company. It is no surprise that this approach is far more robust and gives far better results (upto 91% directional accuracy) than any previous work.

References

1. Acemoglu D, Ozdaglar A, ParandehGheibi A (2010) Spread of (mis)information in social networks. Games Econ Behav 70(2):194–227
2. Asur S, Huberman BA (2010) Predicting the future with social media. Computing 25(1):492499
3. Bagnoli M, Beneish Messod D, Watts Susan G (1999) Whisper forecasts of quarterly earnings per share. J Account Econ 28(1):27–50
4. BBC News (2011) Twitter predicts future of stocks, 2011. This is an electronic document. Date of publication: 6 Apr 2011. Date retrieved: 21 Oct 2011. Date last modified: [Date unavailable].
5. Bollen J, Mao H, Zeng X (2011), Twitter mood predicts the stock market. Computer 1010(3003v1):1–8.
6. Boyd DM, Ellison NB (2007) Social network sites: definition, history, and scholarship. J Comput-Mediated Commun 13(1):210–230
7. Brown JS, Duguid P (2002) The social life of information. Harvard Business School Press, Boston
8. Da Z, Engelberg J, Gao P (2010) In search of attention. J Bertrand Russell Arch (919).
9. Das SR, Chen MY (2001) Yahoo! for Amazon: sentiment parsing from small talk on the Web. SSRN eLibrary.
10. Dewally M (2003) Internet investment advice: investing with a rock of salt. Financ Anal J 59(4):65–77

11. Doan S, Vo BKH (2011) Collier N (2011) An analysis of Twitter messages in the 2011 Tohoku earthquake. ArXiv e-prints, Sept
12. Frank MZ, Antweiler W (2001) Is all that talk just noise?. The information content of internet stock message boards, SSRN eLibrary
13. Garman MB, Klass MJ (1980) On the estimation of security price volatilities from historical data. J Bus 53(1):67–78
14. Gilbert E, Karahalios K (2010) Widespread worry and the stock market. Artif Intell:58–65.
15. Go A, Bhayani R, Huang L (2009) Twitter sentiment classification using distant supervision.
16. Guresen E, Kayakutlu G, Daim TU (2011) Using artificial neural network models in stock market index prediction. Expert Syst Appl 38(8):10389–10397
17. Lee KH, Jo GS (1999) Expert system for predicting stock market timing using a candlestick chart. Expert Syst Appl 16(4):357–364
18. Lerman A (2011) Individual investors' attention to accounting information: message board discussions. SSRN eLibrary.
19. Malkiel BG (2003) The efficient market hypothesis and its critics. J Econ Perspect 17(1):59–82
20. Mao H, Counts S, Bollen J (2011) Predicting financial markets: comparing survey, news, twitter and search engine data. Quantitative finance papers 1112.1051, arXiv.org, Dec 2011.
21. McIntyre D (2009) Turning wall street on its head, 2009. This is an electronic document. Date of publication: 29 May 2009. Date retrieved: 24 Sept 2011. Date last modified: [Date unavailable].
22. Miao H, Ramchander S, Zumwalt JK (2011) Information driven price jumps and trading strategy: evidence from stock index futures. SSRN eLibrary.
23. Qian B, Rasheed K (2007) Stock market prediction with multiple classifiers. Appl Intell 26:25–33
24. Qiu L, Rui H, Liangfei, Whinston A (2011) A twitter-based prediction market: social network approach. In: ICIS Proceedings. Paper 5.
25. Rao T, Srivastava S (2012) Analyzing stock market movements using twitter sentiment analysis. Proceedings of the 2012 international conference on advances in social networks analysis and mining (ASONAM 2012), ASONAM '12. DC, USA, IEEE Computer Society, Washington, pp 119–123
26. Sprenger TO, Welpe IM (2010) Tweets and trades: the information content of stock microblogs. SSRN eLibrary.
27. Szomszor M, Kostkova P, De Quincey E (2009) swineflu : Twitter predicts swine flu outbreak in 2009. 3rd international ICST conference on electronic healthcare for the 21st century eHealth, Dec 2009.
28. The Atlantic (2011) Does anne hathaway news drive berkshire hathaway's stock? This is an electronic document. Date of publication: 18 Mar 2011. Date retrieved: 12 Oct 2011. Date last modified: [Date unavailable].
29. Tumasjan A, Sprenger TO, Sandner PG, Welpe IM (2010) Predicting elections with twitter: what 140 characters reveal about political sentiment. In: International AAAI conference on Weblogs and social media, Washington DC, pp 178–185.
30. Wysocki P (1998) Cheap talk on the web: the determinants of postings on stock message boards. Working paper.
31. Zeledon M (2009) Stocktwits may change how you trade. This is an electronic document. Date of publication: 2009. Date retrieved: 01 Sept 2011. Date last modified: Date unavailable.
32. Zhang X, Fuehres H, Gloor PA (2009) Predicting stock market indicators through twitter i hope it is not as bad as i fear. Anxiety, pp 1–8

The Impact of Measurement Time on Subgroup Detection in Online Communities

Sam Zeini, Tilman Göhnert, Tobias Hecking, Lothar Krempel and H. Ulrich Hoppe

Abstract More and more communities use internet based services and infrastructure for communication and collaboration. All these activities leave digital traces that are of interest for research as real world data sources that can be processed automatically or semi-automatically. Since productive online communities (such as open source developer teams) tend to support the establishment of ties between actors who work on or communicate about the same or similar objects, social network analysis is a frequently used research methodology in this field. A typical application of Social Network Analysis (SNA) techniques is the detection of cohesive subgroups of actors (also called "community detection". We were particularly interested in such methods that allow for the detection of overlapping clusters, which is the case with the Clique Percolation Method (CPM) and Link Community detection (LC). We have used these two methods to analyze data from some open source developer communities (mailing lists and log files) and have compared the results for varied time windows of measurement. The influence of the time span of data capturing/aggregation can be compared to photography: A certain minimal window size is needed to get a clear image with enough "light" (i.e. dense enough interaction data), whereas for very long time spans the image will be blurred because subgroup membership will indeed change during the time span (corresponding to a moving target). In this sense, our

S. Zeini (✉) · T. Göhnert · T. Hecking · H. U. Hoppe
University Duisburg-Essen, Duisburg, Essen, Germany
e-mail: zeini@collide.info

T. Göhnert
e-mail: goehnert@collide.info

T. Hecking
e-mail: hecking@collide.info

H. U. Hoppe
e-mail: hoppe@collide.info

L. Krempel
Max-Planck-Institute for the Study of Societies, Cologne, Germany
e-mail: krempel@mpifg.de

F. Can et al. (eds.), *State of the Art Applications of Social Network Analysis*, Lecture Notes in Social Networks, DOI: 10.1007/978-3-319-05912-9_12,

target parameter is “resolution” of subgroup structures. We have identified several indicators for good resolution. In general, this value will vary for different types of communities with different communication frequency and behavior. Following our findings, an explicit analysis and comparison of the influence of time window for different communities may be used to better adjust analysis techniques for the communities at hand.

1 Introduction

With the rise of web based social media, more and more profession-al and hobby communities use internet based services and infrastructure for communication and collaboration. All these communities leave traces that are of high interest for research as easily accessible and processable real world data. Social network analysis [18] is one of the preferred research methodologies in this field, especially be-cause structural analyses can be computed easily.

From an SNA perspective, the shift from classical network survey techniques to automated capturing of data from network sources has opened new opportunities but has also raised new methodological is-sues. One of these issues is the adequate and explicit handling of time. On the one hand, communication data from networked communities comes with time stamps as an explicit attribute. Indeed, we find numerous studies of the dynamics of networks in the sense of evolution over time (see next section), yet the effect of the time span of data aggregation and measurement is usually not explicitly analyzed.

We came across the problem of window sizes in the context of studying positional role models to identify potential innovators in productive networked communities, particularly open source soft-ware developers. As data sources we used the freely available mailing lists (communication) and software archive log files (cooperation). In general, we were able to show that network positions and their evolution over time can be used as indicator for potential innovators in communities [11]. A further refinement following the distinction between individual innovators versus groups of innovators led us to a comparison between the classical individual broker model [4, 9] and analyses involving overlapping members between k-cliques in the sense of group brokerage [16]. While using the Clique Percolation Method (CPM) [13] to identify overlapping members be-tween clusters we found that different sizes of time slices for data aggregation led to considerably different clusters. For instance, selecting a time slice of one or two years for a very active source code repository would lead to one big clique including practically all members of the network while selecting time slices in the length of days in highly specialized developer mailing lists may only render very sparse and arbitrary connections.

In general, the influence of the time span of data capturing/aggregation can be described using a photographic analogy: A certain minimal window size is needed to get a clear image with enough “light” (i.e. dense enough interaction data), whereas for very long time spans the image will be blurred because subgroup membership

will indeed change during the time span (moving target). In this sense, we strive for optimizing the choice of the target parameter "resolution" with respect to subgroup structures.

For familiar communities, researchers usually have an idea of an adequate length of time slices under the specific conditions. This is often defined by interest in different types of affiliations, e.g. 14-days window length as middle-term relationship between observed student interactions and so on. So there is an implicit idea of inherent paces or fitting time spans existing in given productive groups or communities of which researchers are aware while sampling their cases. For unknown large communities this kind of implicit assumption is missing and the length of time windows is chosen by contingency or comparability between cases.

This leads us to the idea that an adequate size for time slices will lead to ranges that represent the inherent pace of community and will thus yield meaningful clusters. After introducing some related work, in this book chapter, based on our previous work [19], we will discuss the size of time window slices as a determining factor for resolution in the detection of clusters using the Clique Percolation Method and the Link Communities Approach on communication data from open source projects. Finally, after describing and analyzing the empirical cases, we will discuss possible applications and extensions of this approach for determining adequate time slice sizes adjusted to an authentic tempo for different communities regarding to their productivity cycles.

2 Related Work

There is a considerable body of research regarding the dynamics of networks according to evolution and growth. In her recent Ph.D. thesis, Falkowski [8] has developed and applied clustering techniques to detect communities in social networks and to determine the evolution of these structures over time and gives a good overview on the general literature on dynamic networks but does not take the effect of variation of the size of time slices into account. Also the fact that the group structures as perceived by the actors in a community and group structures inferred from network data may differ and therefore the size of time slices has to be considered as an influence factor had already been mentioned once in the early 1980s [2]. However, this problem has not been addressed by community researchers later. We assume that for nowadays? huge online communities it is even more important to determine a time window size that is appropriate for observing the group dynamics, especially if network data are com-pared to cognitive self-perceptions. On platforms like facebook and LinkedIn it is not unusual for one individual member to have declared friendship relations with more than 500 other members alt-hough this number renders the strength or significance of these single ties highly questionable (cf. Dunbars number [7]).

Other relevant approaches dealing with problems in temporal network dynamics are moving structures [14], in which the overlap between time slices is considered. Also there is other work questioning the meaningfulness of static measures like

centrality for dynamic networks [15]. Issues similar to the problem of finding out best fit-ting window lengths occur in the field of network text analysis concerning the right size for window lengths over text fragments to determine the co-occurrence between concepts [6].

Another portion of related work uses CPM in context of temporal dynamics of communities. Wang and Fleury [17] propose a frame-work addressing the influence of temporal variation by "regularizing communities by initializing sub-communities corresponding the cores and proportionality". Palla et al. [12] formulate an extension on CPM by referring to size of communities and their age as heuristic. Finally Greene et al. [10] develop a model for tracking communities in dynamic social networks based on events. A recent paper addressing concretely the identification of best fitting window sizes in order to increase link prediction accuracy was published by Budka et al. [3] at about the same time as our initial research [19]. Last but not least the survey paper by Coscia et al. [5] contains a good overview on the analysis of dynamic communities.

3 Approach

In contrast to several other standard techniques, the Clique Percolation Method (CPM) as described by Palla et al. in [13] allows for identifying cohesive subgroups or communities in social networks with overlapping members between different clusters, whereas many other methods only yield disjoint clusters. As in open source projects, cohesive subgroups may correspond to project teams and project coordinators or senior advisors would usually be involved in several of those subgroups, CPM was the approach chosen for this case study. A community in the sense of CPM is defined as "the union of all k-cliques that can be reached from each other through a series of adjacent k-cliques" [13]. In this definition k-cliques are complete subgraphs of size k and two k-cliques are adjacent if they share $k - 1$ members. The term "percolation" in this case refers to the k-clique construct that percolates through a graph over the adjacent k-cliques with only one node moving at a time (see Fig. 1 the upper row shows the original graph and the community found with 4-clique percolation and the lower row shows the percolation of the 4-clique through the graph).

Similar to CPM, the Link Communities (LC) method [1] also detects overlapping subgroups. The difference is, however, in the interpretation of overlaps. The notion of a community is completely different from other methods. Groups are defined as social dimensions or contexts (e.g. common interest, friendships, etc.) with hierarchical structure and pervasive overlaps, rather than the common definition as parts of networks with more internal than external links. Whereas nodes are usually involved in several contexts, links are assumed being related to a single context only. To capture the hierarchical and massively overlapping organization of groups, the method performs a hierarchical clustering of links instead of nodes. The similarity of two edges with one common incident node is based on the similarity of their neighborhoods. Edges without common incident nodes are considered as not being similar.

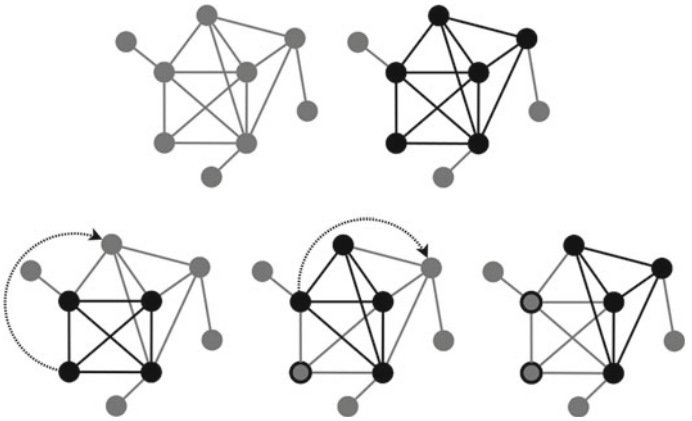


Fig. 1 Percolation of a 4-clique

Using time stamped data, the network data can easily be sliced into different window sizes. For the cases described here, the data was collected from the mailing lists and the log files of the source code version management system of an open source development communities over two years. Especially in the case of the version management data the problem became very clear. This kind of data can be described as very dense in the sense of events as there are a lot of changes in the code done by few "maintainers" having permissions to commit code to the repository. While there are cliques in monthly and quarterly slices, the community becomes a big clique when viewed in the whole time of two years. The same tendency is also observable for the developer mailing list. The intuitive choice of monthly or quarterly time slices which was the basis of our earlier analyses led to meaningful results but also brought us to the question of determining the optimal size for time slices assuming that every community has its own productive speed and inherent time.

We identified several indicators, which reflect the group structure of networks and can be used as basis for a mechanism that supports finding the ideal time slice for a given network. For our application we define an indicator as a function $i(N, D) \rightarrow r$ which maps the network N and the group structure found by the subgroup detection method D to a real value r. The basic indicators we use are number of clusters, aggregated forms of cluster size, coverage, and overlap coverage. Coverage and overlap coverage are defined similarly by the ratio of number of nodes being in at least one cluster (coverage cov) or being in at least two clusters (overlap coverage ol_cov) to the number of nodes in the network:

$$cov(N, D) = \frac{|\{v \in V(N) : |\{\exists c \in C(N, D), v \in c\}| > 1\}|}{|V(N)|} \tag{1}$$

$$ol\_cov(N, D) = \frac{|\{v \in V(N) : |\{c : c \in C(N, D), v \in c\}|\}|}{|V(N)|} \tag{2}$$

Table 1 Candidates for objective functions

	Maximum	Average	Standard deviation
Number of clusters	max num	avg num	sd num
Maximal cluster size	max max_size	avg max_size	sd max_size
Average cluster size	max avg_size	avg avg_size	sd avg_size
Coverage	max cog	avg cov	sd cov
Overlap coverage	max ol_cov	av ol_cov	sd ol_cov
Combined indicator	max ci	avg ci	sd ci

In both of the above formulas, $V(N)$ is the vertex set of a network N and $C(N, D)$ is the set of clusters found by subgroup detection method D applied to network N.

In contrast to the other indicators, cluster size needs to be aggregated before it can be used as a measure describing a single network. As aggregation forms we use maximum, average, and variance or standard deviation. Each of these aggregations gives one single value for a single network. Thus the aggregated forms of cluster size qualify as basic indicators.

In addition to these simple indicators we introduce a combined indicator based on the number of clusters and the overlap coverage. For the case $|\{c : c \in C(N, D)\}| > 1$ it is defined as:

$$ci(N, D) = \frac{|\{c : c \in C(N, D)\}| \cdot m}{|V(N)|} + ol\_cov(N, D) \tag{3}$$

and as $ci(N, D) = 0$ in any other case. Here, m denotes the minimal cluster size. We have introduced it to allow the comparison between CPM and other subgroup detection methods. For CPM the parameter k denotes the size of the clique being percolated through the network and thus determines the minimal size of clusters being found. For comparing the results of CPM with methods which do not restrict the minimal cluster size, like LC, we set $m = k$ and ignore all clusters of a size smaller than m. If not stated otherwise we have used $k = m = 4$ throughout this work. We need the case distinction because if we have only one cluster the outcome would be simply the inverse of the number of nodes, which does not state anything about the subcommunity structure.

Based on these indicator functions objective or target functions can now be defined. These objective functions shall have the form $o(T, D) \to r$ with T being a time series; a series of time slices (networks) of the same length taken from the same community. To transform our basic indicators into such an objective form we need to aggregate again, this time over all networks in a time series. For the purpose of aggregation we again use the same forms of aggregation as for the aggregation of cluster size into a measure for one network. The maxima of such an objective function should now point to time window sizes (here represented by time series), which allow insights into the dynamics of the community in question. All in all we come up with the following candidates for objective functions (Table 1).

We have used both standard deviation and variance as possible aggregation methods but as those are closely related we have only included standard deviation in the table of candidates for objective functions.

4 Case Studies

We chose the open source projects Asterisk, OpenSimulator (OpenSim), and the Dojo toolkit to test our assumptions and evaluate and apply our approach. Each of the projects defined as case studies represent a different domain and also a different type of community in respect to size and frequency of actions.

Asterisk is a software based telephone private branch exchange (PBX) developed by Digium. It is released under the GNU public license as well as under a proprietary license (dual license model). There is a big community working on Asterisk. The software based telephone PBX is often used within call-centers and for this purpose open source is a good choice since the companies can change and adapt the software in respect to their individual needs. Asterisk represents a typical big and productive community with a more or less stable core of developers and users as well as a high fluctuation of occasional users. In general we have collected data from the developer mailing list, the community mailing list and also from the logfiles of the source code version management system (svn) using our data-multiplexer-demultiplexer approach and tool [11]. In case of Asterisk the data covers the whole years 2006 and 2007. The developer mailing list contains 13,542 messages in 4,694 threads discussed by 1,324 developers, the community mailing list 67,949 mails in 26,095 topics discussed by 4,642 contributors, and the svn data 17,868 revisions of 1,866 artifacts from 30 developers. OpenSimulator is a project developing an open source server for hosting 3D simulation worlds, comparable to Secondlife®. It is released under the BSD License. The community is somehow similar to Asterisk, there are maintainers, developers, hosters and end users. It is much smaller though. The data covers the time period of September 2007 to February 2009. The developer mailing list contains 5,505 messages in 1,185 threads discussed by 198 developers, the community mailing list 1,582 mails in 634 topics discussed by 175 people, and the svn data 32,867 revisions of 6,012 artifacts from 26 developers. OpenSimulator represents a typical mid-size open source community with a relative stable core of developers and regular users and some occasional users. The Dojo toolkit is an open source Javascript library developed by the Dojo Foundation. It started in 2004 and the framework is targeting needs of client side web development. It is released under a modified BSD License as well as under the Academic Free License. The data for the mailing list covers the complete years 2006 and 2007. The svn data contains whole 2006 and January to August 2007. The mailing list contains 7,207 messages in 1,477 threads discussed by 114 developers. The svn data contains 15,845 revisions of 4,151 artifacts from 29 users. Dojo represented at the time of observation a typical small-size open source project with a stable core of developers and users but only few occasional users.

To evaluate our approach for the adequate identification parameters in terms of window sizes and clique orders to get an optimal resolution of the cohesive subgroup structure in a given community we decided to use the data from the developer mailing lists because it is less formally pre-regulated than repository access and more open for new developers so that cliques are more likely to emerge and change over time. The opposite assumption that pre-regulated networks of maintaining developers with write access rights based on their svn contributions will tend to build one large clique over time has been tested and confirmed for all three cases with each of our approaches. For this reason we focus to describe and discuss our approach only on the networks derived from the developer mailing lists. Regarding these lists OpenSimulator and Dojo are quite similar projects in the sense that both have between 100 and 200 active members of these lists and between 5,000 and 7,500 contributions to these lists over the observation period. Asterisk is much bigger with more than 1,000 members and more than 13,000 contributions on the mailing list. But there is also a qualitative difference between OpenSimulator and Dojo. Dojo tends to have little fluctuation within the whole developer community whereas OpenSimulator has a stable core of developers but also high fluctuation in the periphery.

5 Analysis

The initial assumption was that evolving networks bear an inherent time which reflects the changes in the community structure. This inherent time might not be stable over the entire period of observation. Moreover highly dynamic networks emerge through different phases [20]. In the email networks of open source communities the communication between the members might rise, e.g. in phases short before a release date.

5.1 Overall Characterization of the Community Networks

To further examine the effect of the observation period on the choice of proper time window sizes we have to have a closer look on the evolution of the particular networks. The upper two diagrams of Fig. 2 show the evolution of the density and the number of nodes of the Asterisk network over a period of 720 days split up in monthly time windows. It is easy to see that this network does not evolve uniformly. The network density increases over the time period whereas the number of nodes decreases. Especially interesting are irregularities like the changes in the number of participating actors within the period between month 4 and month 10. Within this period number of nodes and the density of the network develop in a similar manner. Both decrease in the first two months of this period but later also increase in parallel. This means that together with the new (or returning) members of the network more connections per node are introduced than the average number of connections a node

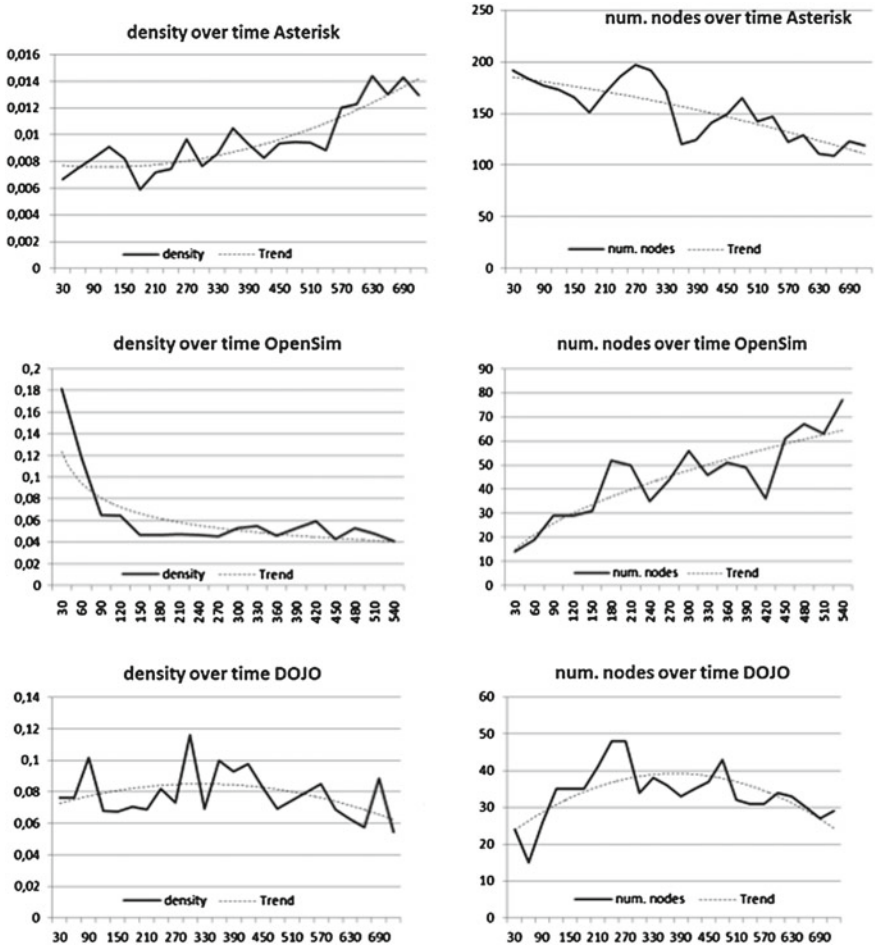


Fig. 2 Number of nodes and Density over time for all networks (monthly time slices)

had in the network before their entrance to the network. After one and a half year the density grows sharply with a nearly constant number of nodes. That means that the remaining actors in the Asterisk network intensify their interactions in this phase. Hence the analysis of the dynamic subcommunity evolution of the Asterisk network may benefit from using different time window sizes for the different phases.

The OpenSimulator network shows a growing behaviour. After an initial phase in the first 5 month of observation where the density decreases and the number of nodes increases, the density remains constantly between 0.04 and 0.05 and the number of nodes show a growing tendency. This indicates that this community was in a formation process in the first five month and then the average degree of the nodes increases. In contrast the Dojo network, which is the smallest and densest

network under investigation, seems to evolve more uniformly than the other two. Therefore the complete period of observation can be split up into time windows of equal length. These observations lead to the assumption that the size of time windows should be adjusted to different phases of observation. Another general observation of analyzing dynamic networks over growing time window sizes is that the networks density grows with growing time window sizes and therefore sizes and density of detected subgroups also tends to grow.

5.2 CPM Analysis

Regarding the CPM algorithm this results in the maximal k-value for which clusters can be found increasing with increasing time window size. For the Asterisk and the OpenSimulator developer mailing lists we observed that the highest values of k for which at least one group could be found in every three-month period were $k = 5$ for Asterisk and $k = 6$ for OpenSimulator. As we found in our earlier work [20] that time window sizes of two or three month seem to be promising we decided to focus on the results for k between 3 and 6 to keep these window sizes in the range of candidate window sizes.

Other effects of the growing density of the network over growing time window sizes are expected behaviors of our indicators. The most important one is the effect this has on the number of clusters. While extending the time window there can be two effects here. One effect of the growing density could be the forming of new groups as more members of the network interact with each other and thereby relations between them are formed which increases the number of clusters. The other effect of increasing density could be the merging of groups that decreases the number of clusters. As we assume these possible effects to cause the number of clusters to be first increasing with growing time window size to a maximum as new groups appear and then to decrease with further growth of time window sizes caused by the growing together of groups the number of clusters seems to be the most promising indicator.

Another expected consequence of the growing network density could be observed with the cluster size indicator. One effect of groups growing together would be that there are few groups which are built from large parts of the vertex set of the network and some groups which are small. In the extreme this means that the network is partitioned in one large group and some isolated nodes. This phenomenon has already been observed in earlier analyses of networks with the clique percolation method and leads us to the expectation that a strong increase in the variance of this indicator followed by a decrease to almost zero variance (caused by the fact that only one cluster which comprises almost the complete network can be observed in each time window) indicates that the time window size is too large to allow insights into the network structure.

If all actors were present even at the smallest observed time window size the expected influence of the growing density regarding the coverage indicator would be that the value of this indicator grows with growing time window sizes as the denom-

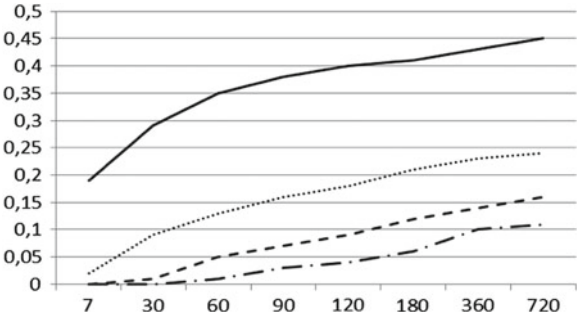


Fig. 3 Coverage for $k = 3$ to $k = 6$ in the Asterisk network (*solid line* is $k = 3$, *dotted line* is $k = 4$, *dashed line* is $k = 5$, and *dashed* and *dotted line* is $k = 6$)

inator of the fraction would be constant and the number of actors that are member of at least one group would grow. If this is not the case then this indicator shows that growing time window sizes also result in an increasing number of actors that are observed and thereby also strengthen the assumption that the observed communities do not only show dynamics in the communication behavior of their actors but also that there is at fluctuation in the group of actors. Figure 3 shows the coverage curves for the Asterisk network. The coverage curves for the OpenSimulator community show similar shapes for the selected k values. Yet, the absolute difference in coverage is big. Given our target of determining active coherent teams in productive online communities, the $k = 4$ curve reaching around 15 % coverage with time slices of 2–3 months and approaching 20 % towards the long end appears to be an adequate choice.

For the whole analysis we assume that some variance in any of the indicator values reflects the fact that the observed network is not static. For both cases the numbers of clusters for each k show a medium range variance, which supports our assumption that this indicator is the most relevant one. Also our expectation regarding the cluster size fits the observation as can be seen in Fig. 4, which leads us to the conclusion that time windows larger than four month do not allow meaningful insights into the structure of the Asterisk network if analyzed with CPM and $k = 3$.

This leads us to going back to using the number of cliques over time for each k, which was the original idea we started with. Since the networks for smaller observation ranges is too sparse for detecting communities, the smallest window length we present is one week. The maximal window size depends on the complete observed time period, it is 720 days for Asterisk and 540 days for OpenSimulator. Both networks show a similar characteristic of the curves.

Figure 5 for example shows that the maximal number of clusters for the OpenSimulator case reaches its maximum in time slices with a length between one and two months for $k = 5$. This means that choosing too short slices like weeks would make the observed results too arbitrary and too long slices like half years would result in too “blurred” and crowded clusters.

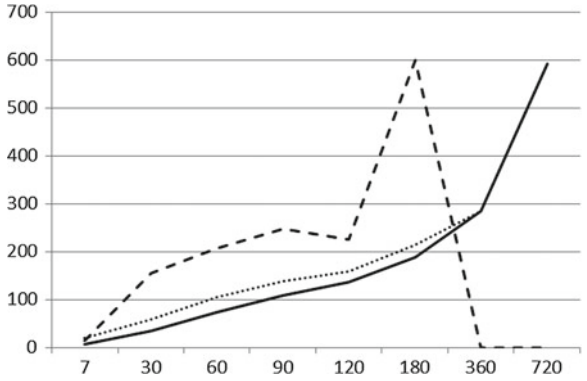


Fig. 4 Cluster size for $k = 3$ in the Asterisk network (*dashed line* variance, *dot line* max. cluster size, *solid line* aver. cluster size)

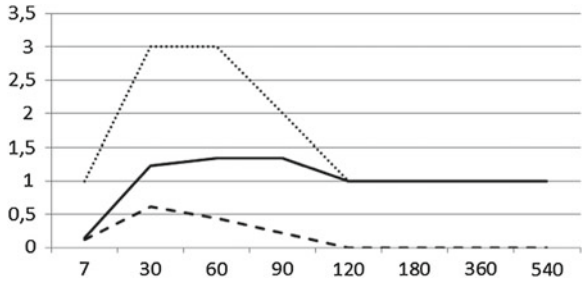


Fig. 5 Maximal number of clusters for $k = 5$ in the case of OpenSimulator (*dashed line* variance, *dotted line* max. cluster size, *solid line* aver. cluster size)

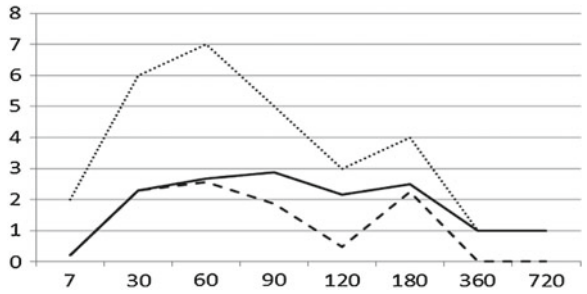


Fig. 6 Maximal number of clusters for $k = 4$ in the case of Asterisk (*dashed line* variance, *dotted line* max. cluster size, *solid line* aver. cluster size)

For the Asterisk network the maximum of the maximal number of clusters as an indicator is more explicit. We assume that this is related to the more hub oriented structure of the network as we can also see by the inverse powerlaw for the degrees. Figure 6 shows this peak at a time window size of two month for $k = 4$.

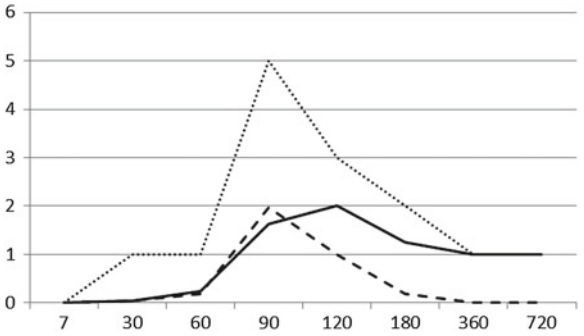


Fig. 7 Maximal number of clusters for $k = 6$ in the case of Asterisk (*dashed line* variance, *dotted line* max. cluster size, *solid line* aver. cluster size)

Our analysis also corroborates our expectation that by choosing longer periods of observation the more distinctive clustering will be found to higher values of k. Figure 7 shows that the maximal number of cliques moves to the right side representing bigger time slices. Compared to $k = 4$ we expect to get clearer insights into the cluster structure by selecting time slices with the length of a three months instead of one to two months.

5.3 Link Community Analysis and Comparison

As an extension of our previous work [19] we compare the CPM approach with the Link Communities approach [1] by analyzing the previously defined indicators graphically with different window sizes to figure out well fitted time windows, so that the subcommunity detection methods produce meaningful results. The aim of longitudinal analysis of subsequent time slices of an evolving network is to understand the dynamics of subcommunities. Therefore it is quite natural to assume that a high variance in the indicator values for a certain time window size indicates that the window size is appropriate to observe the evolution of the subcommunities over the period of observation. The investigated methods are not only different in their subcommunity identification approaches, they differ also in their restrictiveness regarding found subcommunities. The Clique Percolation Method for example requires the specification of a parameter k, which defines the size of the initial cliques that are percolated through the graph. This implies that the smallest possible cluster found by this method is a clique of size k. For Link Communities on the other hand the smallest possible group of nodes is two, the endpoints of a single link. As stated in Sect. 3 we have used $k = 4$ in most cases described in this work and have generally ignored all smaller clusters.

Both the clique percolation method and the link community method are designed for finding overlapping subgroup structures in a network. Therefore it is desirable to

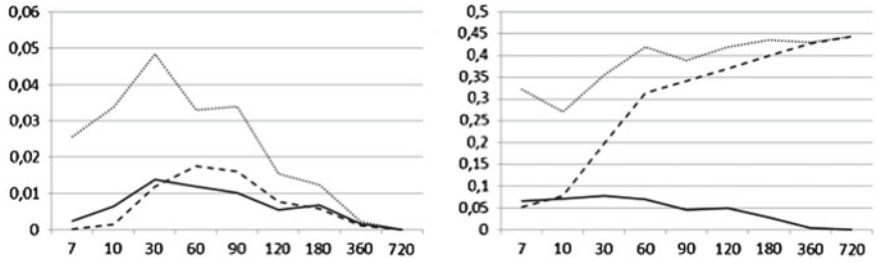


Fig. 8 Overlap coverage for the Asterisk network and different methods CPM (*left*), LC (*right*). (*dashed line* standard deviation, *dotted line* max. cluster size, *solid line* aver. cluster size)

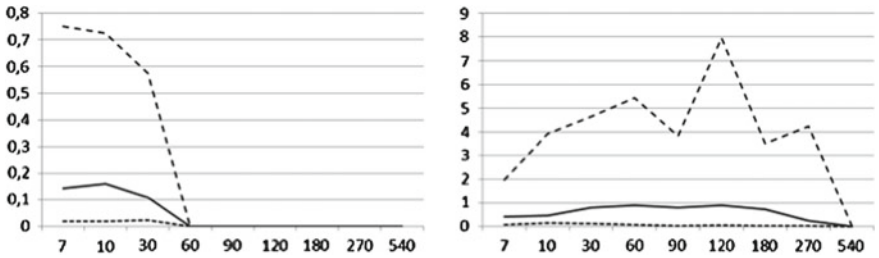


Fig. 9 Combined indicator for OpenSimulator, *left CPM*, *right LC* (*dotted line* std. dev. overlap coverage, *dashed line* std. dev. num. clust., *solid line* std. dev. sum)

investigate the overlap coverage of the clusters found by these methods under different window sizes. Figure 8 depicts the results for overlap coverage in the Asterisk network as an example how the overlap coverage indicator can help figuring out appropriate time window sizes. However, important to note is that the overlap coverage is strongly related to the number of sub communities, because the more sub communities exist the more overlap can occur between them.

The combined indicator introduced in Sect. 3 uses this connection for emphasizing the situations in which both number of clusters and overlap are high. The found results for the OpenSimulator case study are depicted in Fig. 9.

The left diagram shows the results for the clique percolation method, the right for the link community method. In both cases both single indicators and the combined indicator are included. The number of clusters is already in the normalized form as it is used for the combined indicator (see Sect. 3).

The above discussed effect according to growth of density leads us furthermore also to the assumption that the size of time windows should be adjusted to different phases of observation. In Fig. 10 we show an illustrative example of the impact of different observation periods to the number of clusters for our main basic indicators. It depicts the number of clusters that can be identified with the Clique Percolation Method using different time window sizes in the OpenSimulator network.

The diagram on the left corresponds to the first 120 days of observation where the community grows rapidly and on the right to the following 420 days. It appears

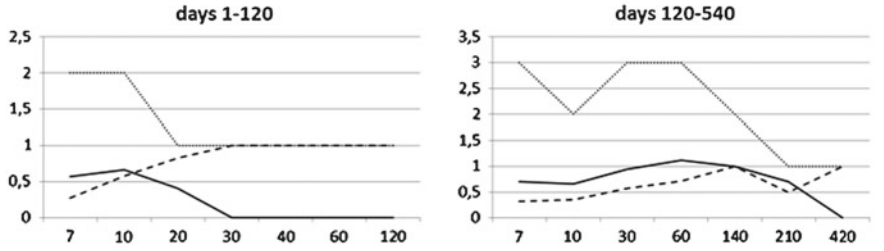


Fig. 10 Number of clusters indicators for CPM in the OpenSimulator network for different observation periods (*left* days1–120, *right* days 120–540, *dotted line* max. num. clust., *dashed line* avg. num. clust., *solid line* std. dev. num. clust.)

that for the first period of observation shorter time window and for the second period longer time windows yield a good resolution of the subcommunity structure. We could observe similar effects with the other methods. As a consequence one can say that more dynamic phases of a network require shorter time window sizes.

Figure 11 shows the number of identified clusters in relation to different window sizes for each method and the networks based on the developer mailing lists of each open source project.

In this chapter only the results regarding the number of clusters and the overlap coverage are presented, because we found that cluster size and coverage are not very useful to determine a proper time window size. These indicators tend to simply increase with larger time window size and consequently don not tell anything about the dynamics of the subcommunity structure.

This analysis of number of clusters shows again the differences between the Asterisk community on the one hand and OpenSimulator and Dojo on the other hand. The OpenSimulator network and the Dojo network are smaller and much denser than the Asterisk network in the sense that more interaction occurs between less people.

This results also in the CPM method not being able to detect more than one giant cluster with time slices larger than three month for OpenSimulator and Dojo. It also clearly shows that the growth in the density of the networks resulting from growing time window sizes has a strong effect on subcommunity detection methods. In contrast to CPM the link community method identifies more clusters when time windows grow as this method initially clusters the edges of a network instead of its nodes. Thus the Link Community method can identify more different clusters in dense networks where many edges can be clustered. As discussed above for the CPM method a peek in the variance of the cluster sizes for particular time window sizes is a good indicator for reasonable time windows. In most cases this raise and decline of the variance/standard deviation can also be observed for Link Communities but it is often not as obvious.

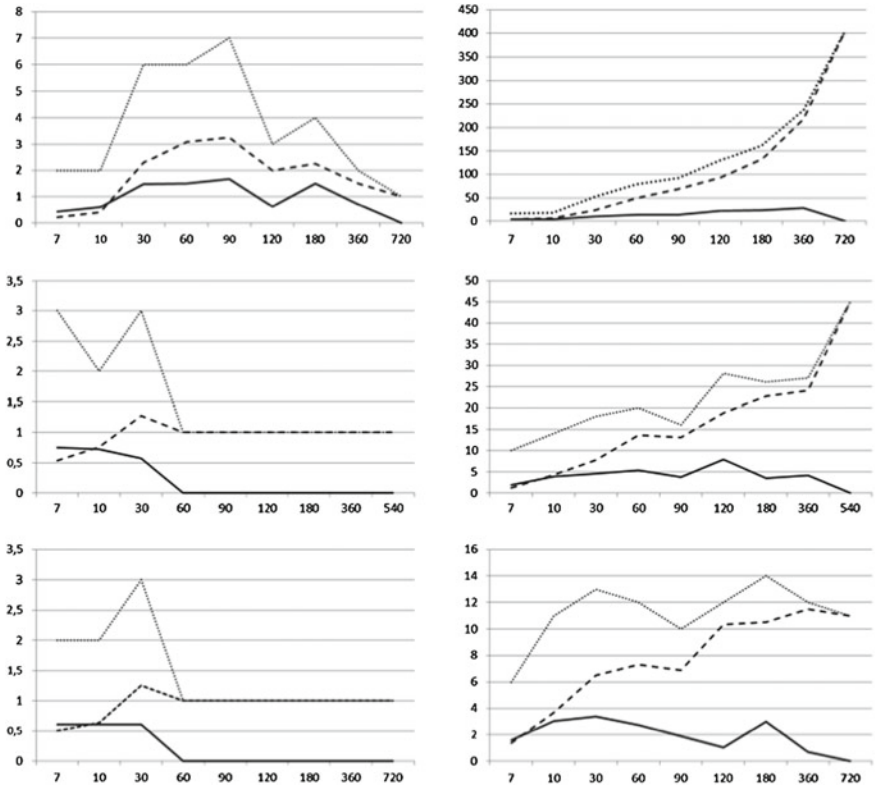


Fig. 11 Number of clusters according to different time window sizes. CPM (*left*), LC (*right*), Asterisk (*up*), OpenSimulator (*middle*), Dojo (*down*) (*dotted line* max. num. clus., *dashed line* avg. num clust., *solid line* std. dev. num. clust.)

Overall and in general these results support our initial intuitive expectation that, for the given cases, time slices of two to three months lead to the best subgroup resolution under the Clique Percolation Method [20].

5.4 Relation Between Community Overlaps and Actor Roles

The extension to more cases and approaches also show that the choice of time windows has to be based on heuristics regarding the specific determinants of the networks. In the productive online communities studied, deadlines and release dates were important determinants of the inherent time structure. The observation that obviously events have an impact of productivity speed in communities leads us to give a closer look at the stories behind the networks to estimate the practical relevance caused by the effect of different time slices.

Table 2 Innovative contributions versus overlapping membership of $k = 4$ cliques in different size time slices

Innovative contributions	7 day slices	30 day slices	120 day slices	Half year slices
Cristina V. L.	Justin C. C.	Justin C. C.	Kyle H.	Dirk H.
Stefan A.	Melanie	Melanie	Dzonatas	Stefan A.
Dahlia T.	Sean D.	Adam F.	David W.	
Dirk H.	Adam F.	Dirk H.		
Justin C. C.	Teravus O.	Dirk K.		
Adam F.	Dirk H.	Stefan A.		
Chris H.	Cristina V. L			
James S.	Dahlia T.			
Kyle G.	Dirk K.			
Michael W.				

The original research question that led us to studying the influence of time window size was related to identifying different brokerage role models and their effects on innovation in productive communities. To reflect this question again in the light of our general findings we focus on the OpenSimulator case. Open Simulator is a quite innovative community. Also the average size and the balanced fluctuations in core and periphery structure indicate it as an ideal network. It is less event driven than Asterisk in phases when support requests dominate the mailing lists. It is also not too much of a "closed shop" like Dojo in some phases, where users tend to act as a big family or clique and everyone does everything.

One of the most innovative topics in OpenSimulator was the idea of so called "Hypergrids". This idea was strongly driven by the computer scientist Cristina V. L., a community member. Her aim was to define a standard protocol similar to hyperlinks in the web, so users in 3D worlds could use these "links" to jump from one simulation world into another. The topic emerged in 6 threads over the time span of 5 months. 5 of 6 threads correlated at least with one of the brokerage role models. In the following we compare the top 10 participants in innovative topics during a quarter in which the Hypergrid topic emerged twice and also correlated with at least one of the brokerage models. We compare them with the top overlapping members at $k = 4$ in different slice sizes. Even though we also find individual brokers (as simple gatekeepers) as well as hubs & authorities, the CPM-based overlapping brokers are indeed the best fitting models in the sense of correlation with participation in innovative topics. So the following comparison (Table 2) is at least tentatively representative for the OpenSimulator community.

Since the slices represent the whole time span and not only the time span of the Hypergrid discussion, it is a quite good indication that Cristina V. L. at least emerges in the top positions of overlapping members in 7 day slices. Most of the names above are heavily involved in OpenSimulator and also mostly having a maintainer position with write permission access to the source code repository. But we also see newcomers like Teravus O. or Dirk K. Dirk for example is a consultant who

was using OpenSimulator indeed as an open innovation project. He did not hold an official role within the OpenSimulator community but his idea of embedding inworld applications (like old 8bit arcade games) into 3D worlds were frequently discussed. Accordingly, he emerges in the 7 days and 30 days slices.

We also have to be aware of formal roles within the communities. Stefan A. and Dirk H. who are the only two overlapping members in the half year slices for example both have important formal positions within the OpenSimulator community. This indicates that formal positions manifest stronger in cliques in bigger time slices whereas freshmen with new ideas occur as broker cliques in shorter time spans. This result underlines our finding with the combined indicator that more dynamic phases of a network require shorter time window sizes.

6 Discussion and Conclusion

Our research shows that a variation of the time window size for data capturing has a systematic effect on subgroup detection. We have studied this using CPM and LC, and first informal trials show similar effects also for other methods, but a more exhaustive systematic exploration of the method-specific characteristics is still on the agenda. We are aware that the underlying notion of community for CPM and LC is quite different and, accordingly, both methods yield different results. However, our point is that both show a systematic dependency of the respective results on the window size. The specific values for most distinctive "high resolution" results are likely to be dependent on the nature of the communities studied. Here not only the size in terms of number of members but also the frequency and distribution of active contributions will likely play a role. We have been looking at "productive communities", which show a high and frequent proportion of active participation as compared to certain hobby or lifestyle communities, in which the share of active members as well as the question/response times are typically much lower. One of the systematic dependencies that we have observed is that there is not one "sweet spot" in terms of a best window size but that, even for one given community, the optimal resolution will depend on the subgroup detection methods and its parameterization.

The proposed indicators offer a starting point for finding combinations of time window sizes and analysis methods that correspond to the "inherent pace" of a network. If we compare, for example, networks based on twitter communication to networks based on scientific co-publication and co-citations is evident that these have different inherent time scales. We believe that in most existing studies, researchers have had an adequate intuition about this inherent pace and the ensuing consequences for data sampling. However, the influence of window size should be determined and characterized to allow for a more explicit and informed handling of time. Also, different characteristics of the dependence of subgroup resolution on time window size and group order may be used to characterize the nature of the communities studied. Pinpointing concrete functions based on what we have shown in this chapter are and will be foundations for techniques and methods that may facilitate awareness

components for collaborative systems and community platforms and to help users as well as deciders to observe and also orchestrate their networks and communities in which they are involved or engaged.

Acknowledgments The authors want to thank to Dan Suthers for the useful comments on the draft version of the chapter. We thank Umar Tarar and Ziyyad Qasim for their contributions to the implementation of the CPM-based analysis. We also thank Evelyn Fricke for the first implementation of the Link Communities analysis used in our study.

References

1. Ahn YY, Bagrow JP, Lehmann S (2010) Link communities reveal multiscale complexity in networks. Nature 466:761–764. doi:10.1038/nature09182
2. Bernard H, Killworth PD, Sailer L (1980) Informant accuracy in social network data iv: a comparison of clique-level structure in behavioral and cognitive network data. Soc Netw 2(3):191–218. doi:10.1016/0378-8733(79)90014-5
3. Budka M, Musial K, Juszczyszyn K (2012) Predicting the evolution of social networks: optimal time window size for increased accuracy. In: 2012 ASE/IEEE international conference on social computing (SocialCom 2012), pp 21–30
4. Burt RS (2004) Structural holes and good ideas. Am J Sociol 110(2):349–399
5. Coscia M, Giannotti F, Pedreschi D (2011) A classification for community discovery methods in complex networks. Statist Anal Data Min 4(5):512–546
6. Diesner J, Carley K (2005) Revealing social structure from texts: meta-matrix text analysis as a novel method for network text analysis. In: Causal mapping for information systems and Te. Idea Group Publishing, Harrisburg (Chapter 4)
7. Dunbar RIM (1993) Coevolution of neocortical size, group size and language in humans. Behav Brain Sci 16:681–694. doi:10.1017/S0140525X00032325
8. Falkowski T (2009) Community analysis in dynamic social networks. Sierke Verlag, G"ottingen
9. Fleming L, Waguespack DM (2007) Brokerage, boundary spanning, and leadership in open innovation communities. Organ Sci 18(2):165–180. doi:10.1287/orsc.1060.0242
10. Greene D, Doyle D, Cunningham P (2010) Tracking the evolution of communities in dynamic social networks. In: Memon N, Alhajj R (eds) ASONAM, IEEE computer society, pp 176–183
11. Harrer A, Zeini S, Ziebarth S (2009) Integrated representation and visualisation of the dynamics in computer-mediated social networks. In: Memon N, Alhajj R (eds) ASONAM, IEEE computer society, pp. 261–266 (2009)
12. Palla G, Barabási AL, Vicsek T, Hungary B (2007) Quantifying social group evolution. Nature 446:664–667
13. Palla G, Derényi I, Farkas I, Vicsek T (2005) Uncovering the overlapping community structure of complex networks in nature and society. Nature 435(7043):814–818. http://dx.doi.org/10.1038/nature03607
14. Stegbauer C, Rausch A (2005) A moving structure: möglichkeiten der positionalen analyse von verlaufsdaten am beispiel von mailinglisten. In: Serdült U, Täube V (eds) Applications in social network analysis (Züricher Politik- und Evaluationsstudien), pp 75–98. doi:10.1007/978-3-531-90420-7_8
15. Trier M, Bobrik A (2007) Analyzing the dynamics of community formation using brokering activities. In: Steinfield C, Pentland B, Ackerman M, Contractor N (eds) Communities and technologies 2007. Springer, London, pp 463–477. doi:10.1007/978-1-84628-905-7_23
16. Vedres B, Stark D (2010) Structural folds: generative disruption in overlapping groups. Am J Soc 115(4):1150–1190

17. Wang Q, Fleury E (2010) Mining time-dependent communities. In: LAWDN—Latin-American Workshop on Dynamic Networks. INTECIN—Facultad de Ingeniería (U.B.A.)—I.T.B.A., Buenos Aires, Argentina, p 4
18. Wasserman S, Faust K (1994) Social network analysis. Methods and applications. Cambridge University Press, Cambridge
19. Zeini S, Göhnert T, Krempel L, Hoppe HU (2012) The impact of measurement time on subgroup detection in online communities. In: The 2012 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM 2012), Istanbul, Turkey
20. Zeini S, Hoppe HU (2010) "Community detection" als ansatz zur identifikation von innovatoren in sozialen netzwerken. In: Meißner K, Engelien M (eds) Virtual enterprises, communities and social networks, Workshop GeNeMe 10. Gemeinschaft in Neuen Medien. TUD press, Dresden

Spatial and Temporal Evaluation of Network-Based Analysis of Human Mobility

Michele Coscia, Salvatore Rinzivillo, Fosca Giannotti and Dino Pedreschi

Abstract The availability of massive network and mobility data from diverse domains has fostered the analysis of human behavior and interactions. This data availability leads to challenges in the knowledge discovery community. Several different analyses have been performed on the traces of human trajectories, such as understanding the real borders of human mobility or mining social interactions derived from mobility and viceversa. However, the data quality of the digital traces of human mobility has a dramatic impact over the knowledge that it is possible to mine, and this issue has not been thoroughly tackled in literature so far. In this chapter, we mine and analyze with complex network techniques a large dataset of human trajectories, a GPS dataset from more than 150 k vehicles in Italy. We build a multiresolution spatial grid and we map the trajectories to several complex networks, by connecting the different areas of our region of interest. We also analyze different temporal slices of the network, obtaining a dynamic perspective over its evolution. We analyze the structural properties of the temporal and geographical slices and their human mobility predictive power. The result is a significant advancement in our understanding of the data transformation process that is needed to connect mobility with social network analysis and mining.

M. Coscia (✉)
CID—Harvard Kennedy School, 79 JFK Street, Cambridge, MA, USA
e-mail: michele_coscia@hks.harvard.edu

S. Rinzivillo · F. Giannotti
KDDLab ISTI-CNR, Via G. Moruzzi, 1, Pisa, Italy
e-mail: rinzivillo@isti.cnr.it

F. Giannotti
e-mail: fosca.giannotti@isti.cnr.it

D. Pedreschi
KDDLab University of Pisa, Largo B. Pontecorvo, 3, Pisa, Italy
e-mail: pedre@di.unipi.it

F. Can et al. (eds.), *State of the Art Applications of Social Network Analysis*,
Lecture Notes in Social Networks, DOI: 10.1007/978-3-319-05912-9_13,

1 Introduction

The availability of massive network and mobility data from diverse domains has fostered the analysis of human behavior and interactions. Traces of human mobility can be collected with a number of different techniques. We can obtain Global Positioning System (GPS) logs, or GSM data referring to which cell tower a cellphone, carried and used by a person, was connecting. The result is a huge quantity of data about tens of thousand people moving along millions of trajectories.

This data availability leads to challenges in the knowledge discovery community. Several different analyses have been performed on the traces of human trajectories. For example, [16, 22] are two examples of studies able to detect the real borders of human mobility: given how people move, the authors were able to cluster different geographical areas in which people are naturally bounded. Another analysis example connects mobility with social networking [4, 25]. The fundamental question in these cases is: do people go in the same places because they can find their friends there or do people become friends because they go in the same places?

However, there is an important issue to be tackled before performing any kind of social knowledge extraction from mobility data. It has been proved that the data quality of the digital traces of human mobility has a dramatic impact over the knowledge that it is possible to mine. For example, in [23] authors perform a trajectory clustering analysis, with GPS data that are successively transformed in GSM-like data. They prove that the knowledge extracted with the semi-obfuscated data is more prone to data noise and performs worse. The conclusion is that mobility analysis should be performed with the high data precision that only GPS is able to provide.

Several open questions are left unanswered, and some of them represent the main focus of this chapter.

The first is connected to the temporal dimension, that is intrinsically linked to any movement data. For example, in [22] authors want to define the borders of human mobility, but they create a rather static snapshot by putting together movements without considering when these movements took place. Also works that consider temporal information usually use it as a continuum without discontinuity points or phase transitions.

In the real world, different events may dramatically change how people move on the territory. Such events may be unpredictable or not frequent, like natural disasters, but most of them are not. The most natural regular and predictable event is the transition between working and non-working days. During Saturdays and Sundays, people usually abandon their working mobility routines for different paths, obeying to completely different criteria. Another example may be organized human social events, like manifestations in a particular town or sport events.

The aim of this chapter is to systematically prove that to mine human mobility and to extract from it useful knowledge is necessary to take into account these phase transitions. A dataset of undifferentiated trajectories, without taking into account when they were performed, may lead to increased and unexpected noise effects, lowering the quality of the results and, in extreme cases, hiding interesting patterns.

The second open question is orthogonal to the temporal dimension and it involves the spatial dimension. Given that we use GPS data, how can we connect it to the territory? In general, GPS does not need to be mapped on the territory, as it already provides the coordinates of the person moving. However, usually we are dealing with two kinds of constraints. First, we are studying vehicles mobility, thus the "data points" are not free to move on a bi-dimensional surface, but they are constrained by the road graph. Second, if we want to apply social network analysis techniques on these data, such as the ones applied in [16, 22] namely community discovery over a network of points in space to find the borders of mobility, we need to discretize the territory in cells, as it is impossible to translate a continuous surface into a graph.

These two considerations force us to discretize the continuous human trajectories into a discrete spatial tessellation and then operate social network analysis on that partition. Should we use external information about the territory, such as the political organization in towns and municipalities? Or should we create a regular grid?

In this chapter, we propose an empirical study aimed at tackling these questions. We collect data from 150 k vehicles moving on a region of Italy, namely Tuscany. First, we address the temporal dimension problem by analyzing with complex network techniques our GPS trajectories and then understand their predictive power of the movements of our observed vehicles over the time span of a month.

Second, we address the spatial dimension problem by creating a multiresolution regular grid that covers Tuscany. We use this grid to generate different network perspectives over Tuscany mobility: grid cells c_1 and c_2 are connected with a directed edge if there is at least one trajectory starting from c_1 and ending in c_2. The edge is then weighted according to how many trajectories connect the two cells.

Both questions are addressed with the same complex network analysis technique, namely community discovery. Community discovery in complex networks aims to detect a graph's modular structure, by isolating densely connected sets of nodes called communities. For the temporal dimension, the communities observed at time t are used to predict the communities observed at time $t + 1$. For the spatial dimension, we verify how well the community partition of a network generated with a particular grid resolution is able to describe the general structure with the minimum amount of information loss.

In the proposed framework, we generate sets of network with different criteria (temporal and spatial). We then apply community discovery on these networks, following our previous works [6, 17], to identify the borders of human mobility. Our focus is to evaluate which temporal perspective and which grid resolution is leading to the best results. We evaluate each network results both quantitatively, using different quality scores, and qualitatively, by looking at the resulting borders and confronting them with what we know about Tuscany mobility.

The rest of the chapter is organized as follows. In Sect. 2 we present the works related to the present chapter: the connections between mobility and social network analysis and mining. We introduce the community discovery problem definition and our adopted solution in Sect. 3. We address our temporal analysis in Sect. 4: we map movements using the political division of the territory, we generated different

temporal slices and we predict the community from one slice to the other. The creation of the multiresolution grid is presented in Sect. 5. Finally Sect. 6 concludes the chapter presenting also some future insights.

2 Related Work

As stated in the introduction, there are several works in the field of human trajectories data mining. A class of these works is focused on applying frequent pattern mining to mobility data [13, 24], even borrowing techniques from biology mining [9]. A popular application to these techniques is the privacy-preserving anonymization of human movements [3, 12]. Different data sources can be used to obtain mobility data ranging from GSM [16], to GPS [17], to RF tags [11]. Sometimes, techniques developed for trajectory mining are then applied in other scenarios [10]. A good taxonomy for mining trajectories can be found in [1].

In literature, there are several works exploring the application of social network analysis to mobility data. Two examples are [16, 22]. In [22] for the first time it is proposed to represent trajectories with a graph, then community discovery techniques are applied to the graph to discover areas that are frequently connected by the same set of trajectories. The mobility data used is the manually submitted information about the movements of one dollar bills in the US territory.[1] In [16] the same approach is implemented, but using GSM cellphone data: each trajectory is composed by the cell tower to which a particular device was connected. As stated in the introduction, the main problems of these approaches is that the data source leads to unavoidable approximations, significantly lowering the quality of the results [23]. We improve over these works by using a more reliable data source, namely direct GPS tracks.

Another class of works is more focused on the links between mobility and social relationships. In [25] a new link prediction technique is proposed. Link prediction in social network is the problem of quantifying how much likely is to observe new connections in a complex network given the current topology of the graph (see for example [19]). The advancement proposed in [25] is to use for the prediction not only the current topology of the graph, but also mobility information about the nodes of the network. The orthogonal problem is tackled in [4]: given the social relationships among a set of individuals, the study aims to predict which trajectories these individuals will decide to take. This class of studies does not focus only on GSM data about real people. In [20], authors focus on movements of virtual spaceships in a massive multiplayer online game, with a wide "universe" to move in. Our chapter is focused on the prerequisites of this class of works, namely how to define the movement graph needed for the analyses.

Finally, as community discovery is used as mean to assess the quality of a network representing human mobility, we report some references about it. Two comprehensive surveys about community discovery are [8], focused on an empirical evaluation of many different algorithms, and [5], that aims to classify the many different

[1] http://www.wheresgeorge.com/

community discovery approaches according to the underlying definition of community they operate on. Several interesting community discovery algorithms are [7, 15, 18, 21], employing different community clustering strategies. We focus particularly on [18], as it is the algorithm we used in the framework presented in this chapter.

This chapter is built on previous work [6]. The focus of [6] was mainly on analyzing the geographical dimension of our problem. We extend over it by introducing a temporal analysis and extended experiments.

3 Community Discovery

An important part of our framework is the application of graph clustering algorithm on our network of trajectories. For this reason, in this section we introduce the problem of community discovery in complex networks along with the solution that we adopted.

An extensive survey, providing more background about community discovery, can be found in [5]. From [5] we know that clustering algorithms can provide extremely different results, according to their definition of what is a community in a complex network. For example, modularity maximization algorithms aim to maximize a fitness function describing how internally dense are the clusters according to their edges. Other techniques use random walks to unveil the modular structure of the network, since the random walker is trapped in denser areas of the network.

When clustering algorithms enable the multi-level identification of "clusters-in-a-cluster", they are defined "hierarchical". With this type of clustering algorithms, we can explore each cluster at several levels and possibly choose the level which, for example, best optimize some fitness function. This is a critical function for mobility networks, as in this scenario it is necessary to explore borders at different granularity levels: conglomerates of cities, cities and even neighborhoods.

Among the hierarchical clustering algorithms available in the literature, we choose the Infomap [18], which is one of the best performing non-overlapping clustering algorithms [8].

The Infomap algorithm is based on a combination of information theoretic techniques and random walks. It uses the probability flow of random walks on a graph as a proxy for information flows in the real system and decomposes the network into clusters by compressing a description of the probability flow. The algorithm looks for a cluster partition M into m clusters so as to minimize the expected description length of a random walk. The intuition behind the Infomap approach for the random walks compression is the following. The best way to compress the paths is to describe them with a prefix and a suffix. Each node that is part of the same cluster M of the previous node is described only with its suffix, otherwise with prefix and suffix. Then, the suffixes are reused in all prefixes, just like the street names are reused in different cities. The optimal division in different prefixes represent the optimal community partition. We can now formally present the theory behind Infomap. The expected description length, given a partition M, is given by:

$$L(M) = qH(Q) + \sum_{i=1}^{m} p_i H(P_i).$$

$L(M)$ is made up of two terms: the first is the entropy of the movements between clusters and the second is entropy of movements within clusters. The entropy associated to the description of the n states of a random variable X that occur with probabilities p_i is $H(X) = -\sum_1^n p_i \log_2 p_i$. In (1) entropy is weighted by the probabilities with which they occur in the particular partitioning. More precisely, q is the probability that the random walk jumps from a cluster to another on any given step and p_i is the fraction of within-community movements that occur in community i plus the probability of exiting module i. Accordingly, $H(Q)$ is the the entropy of clusters names, or city names in our intuition presented before, and $H(P_i)$ the entropy of movements within cluster i, the street names in our example, including the exit from it. Since trying any possible partition in order to minimize $L(M)$ is inefficient and intractable, the algorithm uses a deterministic greedy search and then refines the results with a simulated annealing approach.

4 The Temporal Dimension

In this section we explore the temporal issues of the application of complex network analysis to mobility data. As a proxy of human mobility, we used a dataset of spatio-temporal trajectories of private cars consisting of around 10 M trips performed by 150,000 vehicles. These GPS tracks were collected by Octo Telematics S.p.A., a company that manages on-board GPS devices and data collection for the car insurance industry. Each trajectory is represented as a time-ordered sequence of tuples (id, x, y, t), where id is the anonymized car identifier, x and y are the latitude and longitude coordinates, t is the timestamp of the position. The GPS tracks were collected during a period of one month, from 1st May to 31st May 2011. The GPS device automatically starts collecting the positions when the car is turned on and it stops when it is turned off. The log is transmitted to the server via GPRS connection. Octo Telematics serves the 2 % of registered vehicles in Italy. In our collection, they collected the traces of the vehicles circulating in a bounding box containing Tuscany Region during the period of observation.

To apply complex network analysis on mobility data we first generalize the spatio-temporal positions by means of a spatial tessellation. This is already a challenge *per se*, and we deal more in deep with it in Sect. 5. Since in this section we are focused on the temporal analysis of human mobility networks, we use a simple, sub-optimal, solution. We focus on the origin and destination of each travel of each vehicle. Using the spatial tessellation provided by ISTAT, the statistical bureau in Italy, we associate each origin (destination) to the census sector where the corresponding travel began (ended).

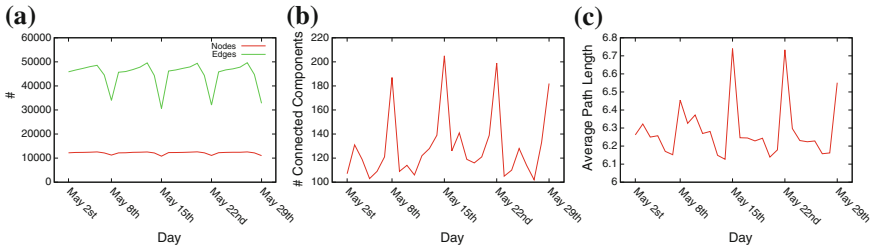


Fig. 1 **a** Number of nodes/edges. **b** Number of components. **c** Average path length. Some statistics for the daily network snapshots

After this generalization step we can model human mobility by means of a graph where the nodes represent the census sectors and each edge represents the set of travels starting and ending within the corresponding census sectors. In particular, an edge connecting the nodes v_1 and v_2 is weighted with the number of travels starting from the sector associated to v_1 and ending at the sector associated with v_2. Moreover, since we are interested in studying the temporal evolution of the extracted network, we extracted several networks at different time intervals. In general, our method consists in selecting only the trajectories "alive" in the time period of study.

Which time interval should be adopted to analyze mobility from a temporal perspective? We fixed a minimum temporal interval of one day and then we generated daily snapshots of the movement graphs. We depict in Fig. 1 some of the basic statistics of these daily networks. We can see that there are remarkable differences between weekday and weekend networks (we recall that May 8th, 15th, 22nd and 29th 2011 were Sundays). Saturdays and Sundays networks usually have less edges, somewhere between 62 and 75 % of the edges of a weekday (Fig. 1a); they have more components, i.e. the networks are more fragmented, with areas not connecting at all to each other (Fig. 1b); and finally their average path length is significantly higher, May 8th presents a lower peak, but the whole preceding week was lower than the following, due to the fact of Italian national holiday of May 1st (Fig. 1c).

We can conclude that we expect different results from the weekdays and weekend networks, as their topology is significantly different. Thus, we considered three distinct intervals for each week: weekdays, i.e. day from Monday to Friday, weekends, i.e. Saturday and Sunday, and the whole week, obtaining 12 networks for the 4 weeks considered.

4.1 Weeks, Weekdays and Weekends Network Statistics

We now take a look to the basic statistics of the extracted networks, as they are able to unveil preliminary differences between the different network views of the dataset. For a deeper explanation about concepts such as "connected component" or "average

Table 1 The average statistics of the different network views of the dataset

Network	$\|V\|$	$\|E\|$	Avg degree	$\|CC\|$	GC size (%)	Reciprocity	ℓ
Weeks	17468.8	218474.0	25.01	20.25	98.8351	0.276039	4.25788
Weekdays	16568.2	167425.0	20.21	26.00	98.7612	0.263951	4.50722
Weekends	13895.5	72055.8	10.37	69.00	97.9868	0.247907	5.33465

path length" we refer to [14]. In Table 1 we reported the following statistics: number of nodes (column $|V|$), number of edges (column $|E|$), average degree (column Avg Degree), number of connected components (column $|CC|$), relative size of the giant component (column GC Size %), reciprocity (column Reciprocity) and average path length (column ℓ). In each row of the table we grouped three kinds of networks: Week, Weekdays and Weekends. Each entry is the average value of the measure of the four networks in each network type.

As we can see, the number of nodes of the Week networks is slightly higher than the number of nodes of the Weekdays networks. This means that during weekends people sometimes choose to reach places that were never visited during weekdays, although in general their destination set is slightly narrower. A big difference between Weekdays and Weekend networks is highlighted by the average degree: during weekends the paths chosen by users are significantly less than what expected by the smaller set of destinations. This means that during weekends the same few paths are under a higher mobility pressure.

Weekends networks appear to be more fragmented (the networks on average present 69 components against the 26 for Weekdays networks), however almost 98 % of destinations are still part of the network's giant component. The giant component size is important because if most of the census sectors are actually isolated from each other, the community discovery loses significance. Also, we know that in Weekends networks we will find 68 very small and isolated communities, that can be ignored for our analytical purposes.

Reciprocity is the ratio of bidirectional edges over the total number of edges. This measure is lower during weekends, implying that in that period of the week people are more likely to stay in the places they reach. Finally, the average path length unveils that we are dealing with classical small-world networks [26]: the average number of edges to be crossed to go from any node to any other node is below 5. An exception is represented again by Weekends networks: although the average path length ℓ is low, it is higher than the other network view, and with a lower number of nodes. We can conclude that the long-range connectivity in Weekends network is weaker than expected.

We depict in Fig. 2 the degree distributions of our 12 networks. We colored in red the Week networks, in blue the Weekend networks and in green the Weekdays networks. The distributions represent an argument in favor of our chosen methodology. The three kinds of networks present very similar degree distributions, while they differ from each other. While the Weekday networks still can approximate the

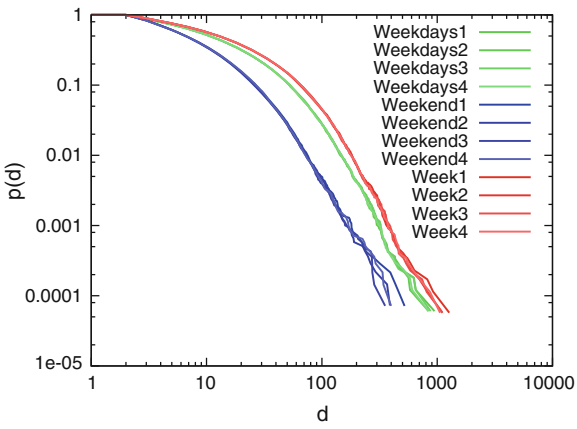


Fig. 2 The cumulative degree distribution of our networks

Week ones, the same does not hold for the Weekend network, that dramatically differ from the previous two. The statement that the Weekend network cannot be useful in predict general patterns of the week, and vice versa, proves to be intuitive. We provide evidences in favor of this statement in Sect. 4.3.

4.2 Evaluation

To evaluate how much the communities discovered in a particular temporal interval are meaningful, we check if they are preserved in different time periods, by comparing each other by means of the measures of precision and recall. We call *clustering* the aggregation of a set of objects into subgroups and each subgroup is called a *cluster*. Formally, a clustering $\mathcal{C}$ is the union of its own clusters $\{C_1, C_2, \ldots, C_n\}$. Given two clusters, say C_1 and C_2, precision and recall are given by the formulas;

$$R(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1|}; \quad P(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_2|}$$

The recall measures how many of the objects in C_1 are present in C_2, while the precision measures the proportion of the object of C_1 in the cluster C_2. The recall of the set C_1 tends to one when all the elements of C_1 are present in C_2, it tends to zero otherwise. The precision of a cluster C_1 tends to zero when the proportion of elements of C_1 is small with respect to the number of element in C_2, and it tends to one when the cluster C_2 contains only elements in C_1.

To extend the measures from the cluster level to the global evaluation of the two clusterings, we propose the following procedure. First, for each cluster C_i in $\mathcal{C}_1$ we determine a cluster $C'_j = map(C_i) \in \mathcal{C}_2$, such that C'_j maximizes the intersection with C_i among all the clusters in $\mathcal{C}_2$. Then, for each pair $(C_i, map(C_i))$ we determine

precision and recall values. The overall similarity indexes is given by the weighted average of each pairs:

$$P(\mathcal{C}_1, \mathcal{C}_2) = \sum_{C_i \in \mathcal{C}_1} |C_i| P(C_i, map(C_i))$$

$$R(\mathcal{C}_1, \mathcal{C}_2) = \sum_{C_i \in \mathcal{C}_1} |C_i| R(C_i, map(C_i)).$$

4.3 Experiments

4.3.1 The Human Mobility Borders: Weekdays Versus Weekends

We start by taking a look at the top-level clusters extracted by the hierarchical version of Infomap algorithm. In Fig. 3 we show a matrix of all the clusterings, for each week and for each network type (Weekday, Weekend and Whole Week). In general, the clusters look mostly compact, with the exceptions of the areas where we can find the highway entrances, as they are of course catalyst hubs of long-range trips. The white areas are regions where no trip started or ended and for this reason are excluded from the network representation. In general, some useful insights can be extracted to improve human mobility management, as the merging of Pisa and Livorno provinces (cluster on the middle-left, light green color in the Weekday map for the week 1, top left corner of Fig. 3): the two cities are divided only for political reasons, but they are very close and part of a strongly connected area, as witnessed by how people move. At least on the higher level, those areas need to coordinate.

In this case, we can exploit the power of the cluster hierarchy to have a finer description of the mobility borders. In Fig. 4 we zoomed into the Pisa-Livorno cluster for the Weekday network of week 1: on the left side we have the cluster at the top level of the hierarchy, on the right side the cluster at the second level. As we can see, at this level the provinces of Pisa and Livorno are correctly split, meaning that there is a border at least at the city level, and our framework is able to detect it by exploring the cluster hierarchy.

Let us now focus on the differences between the Weekdays and the Weekends clusters. The Weekends clusters look as compact as the Weekdays clusters and the *quantity* of differences looks lower than expected from the intuition that the network statistics gave us (see Sect. 4.1). However, the *quality* of the differences is very important: in week 1 the Pisa-Livorno cluster expanded and now includes also the cities of Lucca and Viareggio (black and brown clusters north of Pisa in Fig. 3, respectively), that are naturally separated from Pisa and difficult to reach. The inclusion is probably due to a higher rate of long-range trips to neighboring towns usually difficult to reach, but appealing to spend some free time during the weekend. Also, the Florence cluster (orange in Fig. 3a) is split in two (pink and blue cluster in

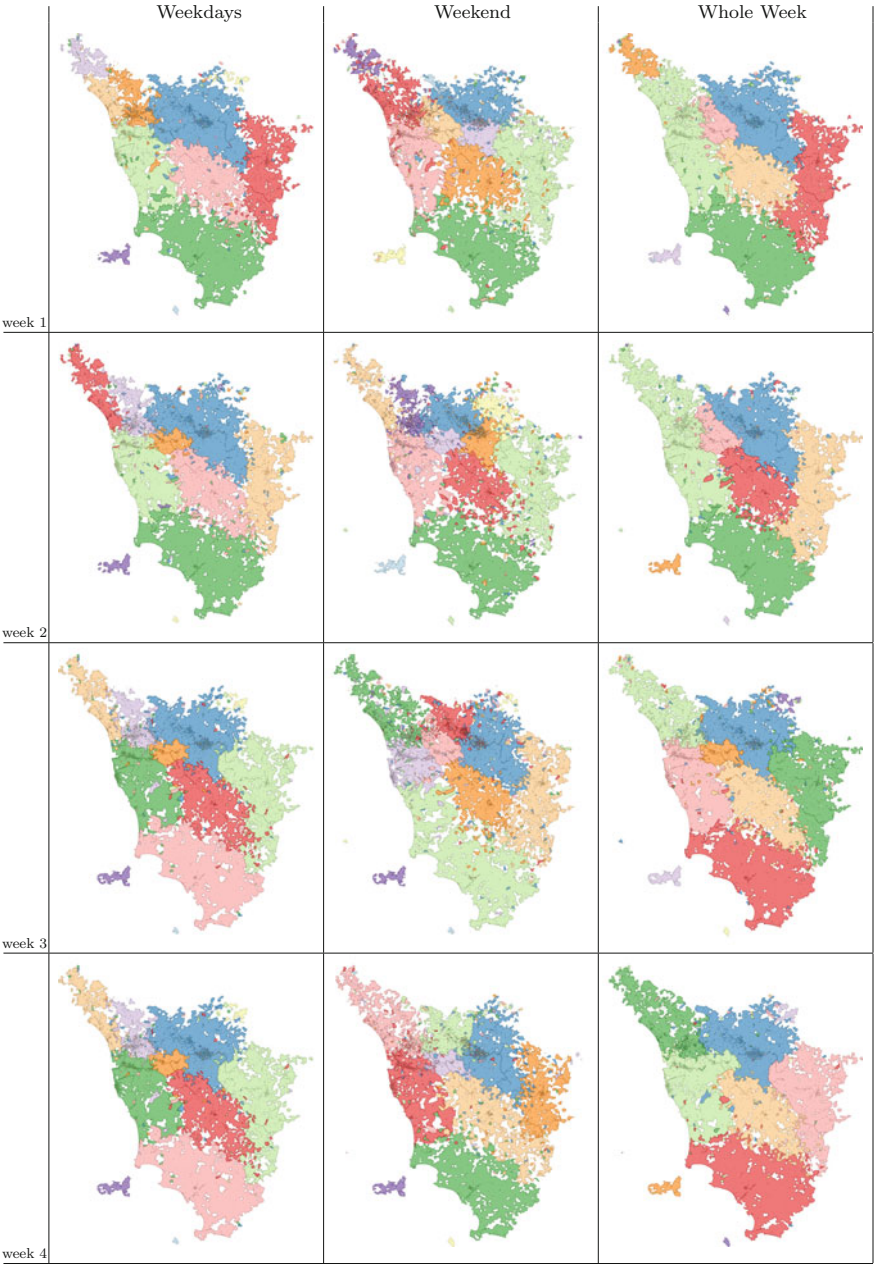


Fig. 3 The Tuscany mobility clusters (top level of the hierarchy).

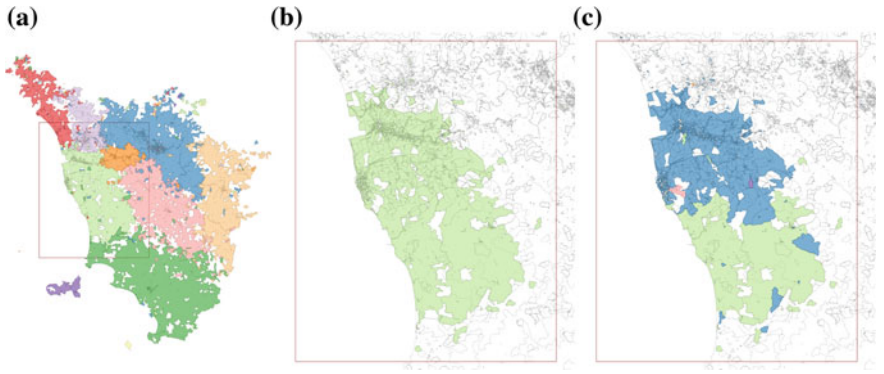


Fig. 4 **a** Weekend 2 clusters at the highest level of hierarchy. **b** Pisa-Livorno cluster at level 1. **c** Pisa-Livorno cluster at the second level of hierarchy. Exploring the second level of hierarchy clusters

Fig. 3b). These changes are very important qualitatively, as these clusters involve a large share of all Tuscany trips. In general, the strong noise effect created by weekend movements is evident for week 3. The Whole Week clusters tend to look in general more alike the Weekdays, but Weekend clusters perturb their borders: the Weekday Lucca cluster (light purple) is split in three parts in the Weekend cluster and this causes its disappearance also from the Whole Week clusters, divided between the pre-existing Florence (blue) and Massa-Carrara-Viareggio (green) clusters. Similar instances of these problems are present in each week, intuitively proving the noisy effect of weekend trajectories.

4.3.2 Weekdays and Weekends Quality Evaluation

We now evaluate the predictive power quality of the cluster extracted from the various networks. We make use of the Precision and Recall measures as defined in Sect. 4.2. The general procedure is the following: we consider the clusters extracted in the network representing the first week and then we calculate the Precision and the Recall for each of the other networks. A high score means that the target network contains similar clustered information, therefore it is predictable using the source network. The results are depicted in Fig. 5.

To understand how to read Fig. 5, let us consider its leftmost scatter plot: in this case the source clustering is calculated using each of the Weekday network. Each dot represent the quality results, according to Precision (x axis) and Recall (y axis), for each of the other network considered in this article. The dot color represent the kind of network to which we are applying the prediction: green for Weekday, blue for Weekend and red for Week. Since we are dealing with four weeks and three different network views for each week (Weekday, Weekend and Week) we have a total of 48 points, 4 of which scores 1 for both Precision and Recall as they are clusterings

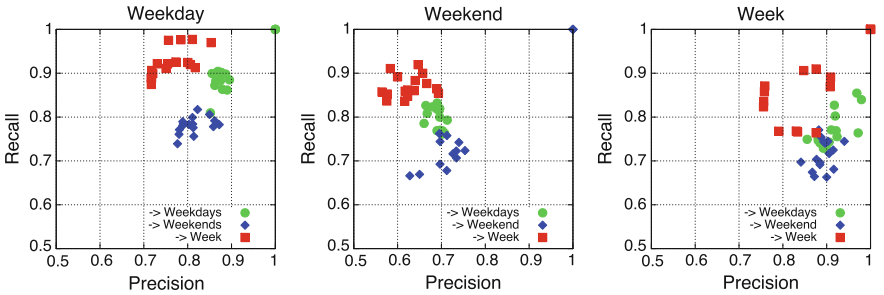


Fig. 5 The precision and recall values for the predictions using weekday (*Left*), weekend (*Center*) and week (*Right*)

applied to themselves: since we are considering the leftmost plot, the 4 perfect scores are all green dots, each representing a Weekday clustering applied to itself.

Now we can find evidences about the lower quality of the Weekend predictions by considering all the three plots. As we can see, the central plot, the one representing the prediction results using the Weekend clusters, scores lower performances for all networks, both in Precision and Recall. Not only Weekend clusterings are not able to predict Weekday and Week clustering: they also score poorly in predicting themselves, proving that from one weekend to another the trajectories vary significantly, and therefore they cannot be predicted efficiently using the simple assumption that the same period in the week should behave in the same way across time.

The other side of the story also holds: not only Weekend cannot predict with high scores, but it also cannot be predicted. By considering the leftmost and the rightmost plot, we see that the distribution of the colors of the dots is not random, but they are clustered in precise areas of the plot. Focusing on the blue dots (Weekend), we notice that they always tend to be clustered in the lower side of the plot, i.e. the one characterized with lower Recall scores. In conclusion, Weekend clusterings are behaving like an unpredictable, and unreliable for prediction, class of phenomena.

However, we also notice that unexpectedly Prediction scores for blue dots in the leftmost and rightmost plots are not the lowest in absolute terms. The explanation lies in the nature of the Week datasets: by definition it also includes the trajectories originated during weekends. This inclusion is lowering the Precision scores for the prediction Weekday to Week and from Week to Weekday. In fact, in the leftmost plot the green dots (Weekday to Weekday predictions) tend also to score better according to prediction, while this does not hold for red dots (Weekday to Week predictions). For the rightmost plot, being the Week-based prediction affected by the weekend data, we have a more confused evaluation. We can conclude that to integrate weekday data with weekend data is equivalent to manually introduce noisy data points, and it should be avoided. It is not true that weekday data can correct the noise of weekend data, or that weekend data can somehow compensate or integrate weekday data. If we want to have reliable models for the majority of human movements, then we should use only weekday data. If we want to have also a model for the irregular human movements during weekends, we need to sacrifice the prediction quality.

4.3.3 Systematic Versus Occasional Trajectories

To evaluate the influence of systematic movements over human mobility, we propose here a method to select the very frequent movements among the travels of each vehicle. Given a vehicle v we select all the travels associated to v and we cluster them according to their starts and ends, i.e. the trips starting from similar places and ending in similar places are aggregated in the same cluster. To extract the clusters from the set of origins and destinations of each vehicle we adopt a density based clustering method, namely OPTICS [2]. OPTICS is one of the best candidates clustering methods since it is very robust to noise, it does discover the natural number of clusters in the dataset analyzed and it can be customized by providing specific distance functions. In our case, we defined a distance function based on the relative distance between origin and destination point of each trajectory. In particular, given two trajectories t_1 and t_2 with end points respectively (s_1, e_1) and (s_2, e_2), the distance between t_1 and t_2 is defined as

$$d(t_1, t_2) = \frac{d(s_1, s_2) + d(e_1, e_2)}{2}.$$

The OPTICS algorithm start exploring the dataset by evaluating the neighborhood of each trajectory according to the distance function provided and to a distance threshold ϵ, which defines a minimum radius around the current object, and a minimum number of point *MinPts* expected to be found within the given radius. When a trajectory has enough neighbors in its radius, it is said to be a core trajectory and its cluster is expanded as far as other density points are reachable. In our experiments we focused on the analysis of very compact clusters that could represent systematic travels. Thus, we used a distance threshold of 250 m. The cluster with the highest cardinality is selected as the most frequent and, hence, as the systematic movement of the vehicle. By repeating this procedure for all the vehicles, we can select a subset of movements that are frequently performed by them. Starting from this subset we apply the same method presented in the previous section: the trajectories are generalized to the spatial tessellation, they are projected in a specific time interval and a complex network is extracted.

In Fig. 6a we report the relative distribution of systematic trajectories in our dataset. For each day, we divide the number of trajectories classified as "systematic" by the number of the total trajectories that were followed during that day. We can see that there is a strong difference between weekdays and weekends. During weekdays, more than 13 % of trajectories are systematic. An exception is Friday, as pre-weekend day, although always at least 12 % trajectories are systematic during that day. During weekends, these shares drop to around 8.5 % during Saturdays and 7.5 % during Sundays. Therefore we can safely state that our assumption, i.e. that systematic trajectories are followed more commonly during weekdays, is sustained by evidence.

The impact of the systematic component of mobility is also evident from the results of community discovering on these networks. Figure 6b show the measures of precision and recall resulting from the comparison of the systematic networks with

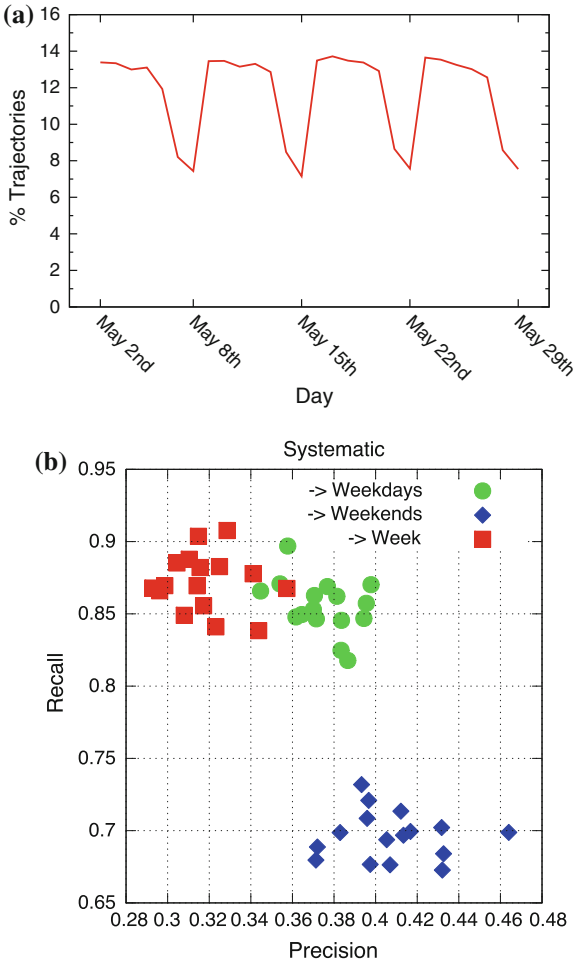


Fig. 6 **a** The daily distributions of systematic trajectories. **b** The quality measure of the communities extracted from the systematic networks. The daily distributions of systematic trajectories: for each day the share of trajectories that are systematic

the networks explored in Sect. 4.3.2. The separation between weekend prediction and week/weekday prediction is here even more evident. In general, the values of recall are very low if compared with Fig. 5. This is due to a sparse network extracted from a limited number of trajectories. We can verify this by looking at the statistics of the networks extracted from the systematic trajectories. In Table 2 we report the same statistics of Table 1, but for the systematic networks instead of the networks created with the complete set of trajectories. We can see that there is an increased number of connected components (ten times more) and the giant component is significantly smaller. Each separated component generates an isolated community, thus greatly

Table 2 The average statistics of the different network views of the dataset, using only systematic trajectories

Network	$\|V\|$	$\|E\|$	Avg degree	$\|CC\|$	GC size %	Reciprocity	ℓ
Weeks	11861.2	26349.8	4.443	240.75	94.7033	0.0290827	7.85268
Weekdays	11059	22748.2	4.11398	269.5	93.8127	0.0270297	8.40738
Weekends	7375.25	8745.5	2.37158	667.75	76.4822	0.0172919	14.4058

lowering the Recall score. The values of precision, in this case, are neatly separated: weekday and week networks maintain similar values, whereas weekend networks have poor prediction performances.

5 The Spatial Dimension

As we saw in the previous section, given the spatial precision of GPS points, it is necessary to process the data in order to generalize neighbor points within a spatial region. Since the spatial precision of a GPS position can have an error of few meters, we need to determine the most suitable generalization for then performing complex network analysis. Our approach consists in studying the properties of a complex network extracted from a regular grid composed of regular squares with edges of the same length.

As a starting point, we consider the bounding box containing our GPS trajectories, i.e. the minimum geographical rectangle that contains all the points, say h and w respectively the height and width of the box. Chosen a length l for the edge of each cell, we divide the bounding box into a grid of cells with r rows and c columns, where $r = \lceil h/l \rceil$ and $c = \lceil w/l \rceil$. The resulting grid is aligned with the lower left corner of the original box.

There are several criteria to partition the territory for a spatial generalization step. In this research, we focus on the spatial resolution of a regular division, since it enables us to control the granularity with a uniform distribution of the cells.

Given a spatial partition, we can extract a network model to represent human movements on the grid. Each travel is mapped to a pair of cells: c_s, the starting cell, and c_e the destination cell. The network is determined by a set of nodes, representing the cells, and a set of edges, representing the travels between two cells. Each edge is weighted with the number of travels connecting the corresponding cells.

By varying the grid resolution as shown in Fig. 7, we are able to generate different network perspective of human mobility, and for each network we can derive basic statistics on its topology. Network basic statistics are an important proxy to understand part of the topology of the network itself. Given the values of measures like average degree or path length, we can understand if the network representation presents a topology that is likely to include a modular structure, thus community discovery can be used effectively.

Fig. 7 (*Left*) A sample of the trajectory dataset used for the experiments. (*Center*) A partition based on a regular grid with cells of size 2,000 m. (*Right*) A partition with a grid with 20,000 m cell size

To refer to distinct granularities, we call each network as "od_net_" followed by the cell size in meters of the underlying grid used to generate the network. Figures 8 and 9 depicts two different sets of statistics. Please note that the figures do not report the absolute value of the particular network measurement, but their relative value w.r.t the value obtained for the network with the largest grid cell, i.e. "od_net_40000". We cannot report the actual values for all networks for lack of space.[2]

Looking at Figs. 8 and 9 we can state some interesting things about the networks generated with different grid resolution levels. First, the number of nodes and edges drops dramatically by passing from a grid size of 200–10,000 m, while sizes greater than 15,000 m do not create much difference. Second, the number of edges drops with a different rate w.r.t the drop in the number of nodes: we can see in Fig. 8 that the green line starts from below the red line, then it is higher in the interval 4,000–17,000 m then drops again. This is consistent to what we see in Fig. 9: the average degree increases until a maximum density for a cell size in between 10 and 15,000 m, then slightly lowers. The average path length drops consistently, while reciprocity and average node weight increase: this is expected as bigger cells includes more trips and it is more probable to have reciprocal edges.

If we want significant results with community discovery we need generally dense networks with small-world properties with not too many small isolated components, and we want to achieve this objective with the smallest possible grid cell, thus with more nodes and edges, to have a more fine-grained description of reality. A preliminary conclusion may be that the optimal cell size should be around 5,000 m: smaller cells generate networks with lower density, or with too many components.

Another important characteristic of the analyzed networks can be observed by when plotting their degree distributions (see Fig. 10). For clarity, we plotted only the degree distributions of the networks generated with a cell size of 500, 1,000, 2,000, 5,000, 10,000, 20,000 and 40,000 m. We can see that all the distributions present a heavy exponential cutoff. However, while the distributions for small cell sizes are

[2] The complete table can be retrieved at the following URL: http://www.di.unipi.it/~coscia/borders/gridstatistics.htm

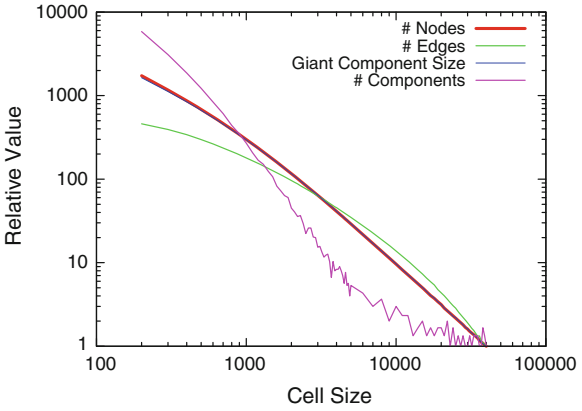


Fig. 8 Some statistics of the extracted networks, relative to the values of the "od_net_40000" network: number of nodes, edges and connected components, and giant component size

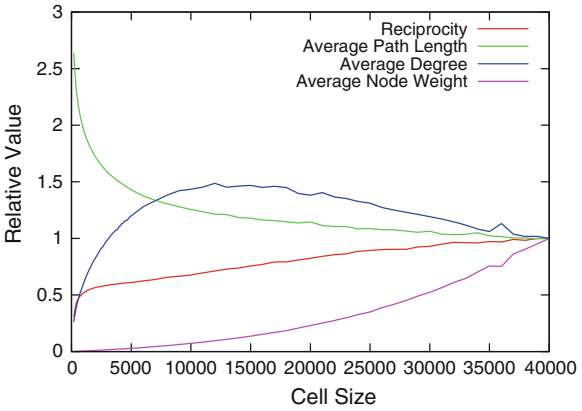


Fig. 9 Some statistics of the extracted networks, relative to the values of the "od_net_40000" network: reciprocity, average path length, degree and node weight

similar, just on different scales, from cell sizes larger than 10,000 m the exponential cutoff is increasingly stronger. This means that networks generated with larger cells lack of a peculiar characteristic of many large complex networks, i.e. the presence of hubs, a set of nodes very highly connected. As their average shortest path is still low, it means that their "small world" properties are not due to the network connectivity itself, but instead to the network small size. Thus, a cell size of 10,000 m seems a reasonable upper bound for the cell size in our dataset. This upper bound can be explained by considering the distribution of lengths showed in Fig. 11: short-ranged travels (up to 10 km) count for the 60 % of the whole dataset. When increasing the grid size, small travels tend to be contained within the same cell, generating a self-link in the resulting network. This reduces the "power" of a cell of attracting other cells in its community, since there are less long-ranged trips.

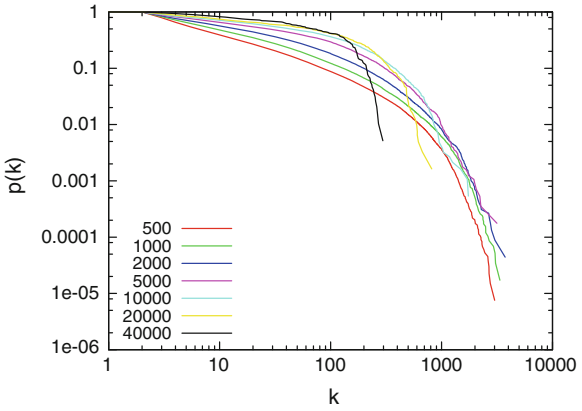


Fig. 10 The degree distributions for the networks generated with different cell sizes

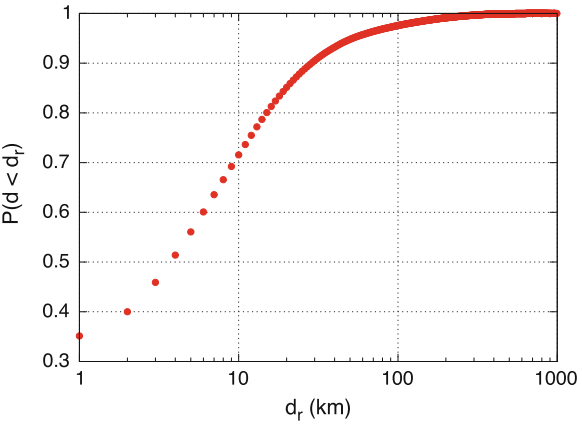


Fig. 11 Cumulative distribution of length of trajectories in our dataset

5.1 Experiments

The communities extracted for each grid resolution are mapped back to the geography and they are used to compute thematic maps of the territory. Given a spatial resolution, for each community we retrieve all the cells associated to its nodes and we join them in a cluster, i.e. a geometric representation of the area covered by the community. An example of such thematic map is presented in Fig. 12. For clarity, areas corresponding to different communities are rendered with different colors. It can be noted the presence of holes in the reconstructed map, since there cells of the spatial partition that do not contains any travel. This phenomenon is more evident for smaller resolutions, where it is possible to find cells that do not contains any road and, thus, any car travel.

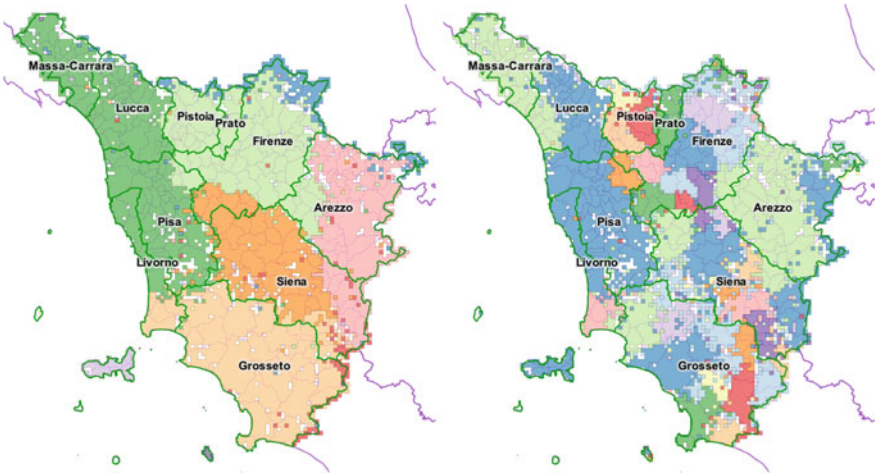


Fig. 12 (*Left*) The clusters obtained with grid cell size of 2,000 m. (*Right*) The clusters determined by the level 2 of the Infomap hierarchy for the same grid resolution

5.1.1 The Borders

We compare the resulting clusters with the existing administrative borders, in particular with the provinces, i.e. an aggregation of adjacent municipalities whose governance has the duty for traffic monitoring and planning. The borders of provinces are drawn with a thick green line in Fig. 12 (*Left*). From the figure it is evident how the emerging communities suggest small variation on the location of the actual borders. For example, the four provinces of Pisa, Livorno, Lucca and Massa are aggregated in a single cluster, since the province of Lucca serves as collector of the mobility of the other three. Exploring the hierarchical aggregation of the communities resulting from Infomap (see Fig. 12 (*Right*)), it is evident the role of the central area of the province, where Lucca is located and where there exists a large vertical cluster (highlighted in blue) connecting the majority of the municipalities of the region. In fact, the cities of Pisa, Lucca, and Livorno form the so-called *area vasta* (i.e. large area), which is characterized by a large flow of commuters. The province of Livorno is divided into two parts, where the north part is included to the province of Pisa and, by transitivity, with the other twos. A similar behavior is observed for the cluster containing the provinces of Firenze, Prato, and Pistoia. These big cities actually form a large metropolitan area with a huge number of commuters moving from one city to the other. This mobility is also sustained by the high capacity of the highway that connects the south with the north through the node of Firenze. The citizens of the city, moreover, have a special reduction for the toll. The provinces of Siena and Arezzo maintain their own borders. It is worth noting that the derived communities follow the borders of each municipality enforcing the internal role of each city as a minimum building block for human mobility borders.

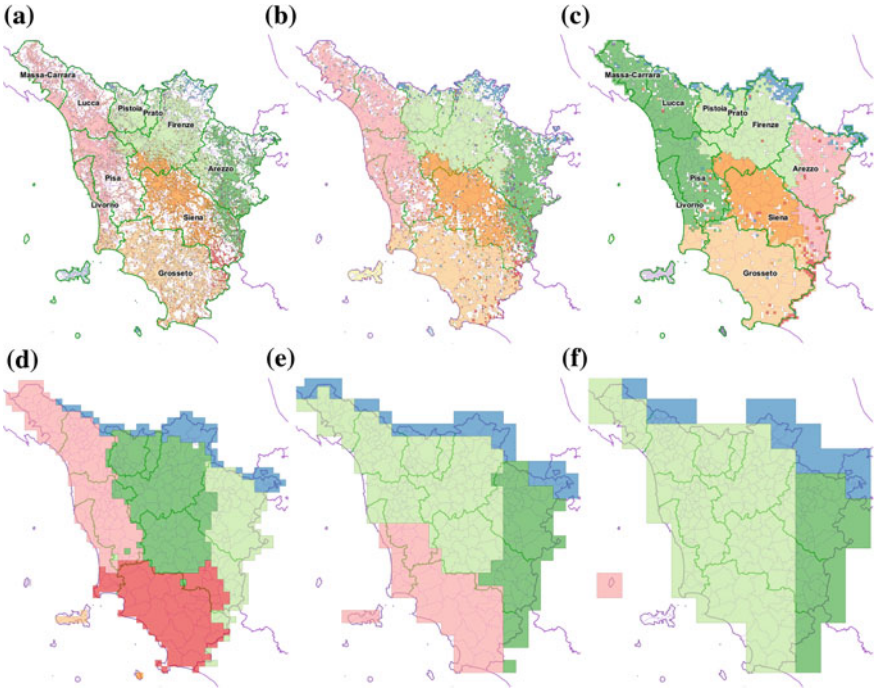


Fig. 13 The resulting clusters obtained with different spatial granularities. **a** 500 m. **b** 1,000 m. **c** 2,000 m. **d** 5,000 m. **e** 10,000 m. **f** 20,000 m

Figure 13 shows the evolution of the clusters at different spatial granularities, namely with size 500, 1,000, 2,000, 5,000, 10,000, and 20,000 m. The first three snapshots show a coherent result, where the clusters identified within the high resolution grid of 500 m are preserved in the successive steps. Starting from a cell size of 5,000 m, the smaller clusters disappear, like for example the cluster between Siena and Grosseto, highlighted in red. When the spatial resolution became more and more coarse, we observe also a merging of distinct clusters in the same communities. In the clusters of resolution 5,000 m, for instance, the cluster of Siena is merged with the cluster formed by Firenze, Prato, and Pistoia. In the other two successive steps the same phenomenon is repeated. At a resolution of 10,000 m the cluster of Firenze is merged with the cluster of Pisa and Lucca. In the coarser version of the grid the resulting clustering actually contains all the grid cells in the same cluster.

From a qualitative evaluation of the resulting maps, we can infer an optimal grid cell size threshold of 5,000 m: smaller granularities allow the identification of reasonable borders at the cost of a more complex computation and with the proliferation of very small local clusters.

5.1.2 Community Quality

Beside a visual comparison with the provinces, we analytically compared the partition derived by the community discovery approach and the partition determined by the actual administrative organization by means of the two measures of *precision* and *recall* introduced in Sect. 4.2. In our setting, for each grid resolution we compare the sets of cells determined by the Infomap algorithm and the set of cells determined by the administrative borders. The administrative borders are represented by the set of grid cells whose cenrtoid is contained within the border interior (we use the centroid of the cell to avoid duplicated cells in different clusters).

The resulting values for precision and recall are plotted in Fig. 14. The plot supports the observation made by means of the visual comparison of the clusters. Recall performs better for smaller grid size, namely up to 2,000 m grid size, it decreases for values between 2,000 and 7,000 m, and it has a drop for larger cell sizes. These results confirm and explain the clusters presented in Fig. 13.

Precision and Recall are not the only evaluation measures we can exploit. Infomap calculates also the code length needed to codify the network given the community partition. Lower code lengths are better because they are the results of a better division in communities. Of course, the simple value of the code length is meaningless in our case, as the networks have very different scales (the number of nodes goes from 335 k to 194 and the number of edges from 4 M to 9 k). Instead, we can adjust the code length with the number of nodes, as it is an information referred to how many bits are needed to represent all the nodes in the network. We adjust the code length with the following formula:

$$CL_{adj} = \frac{CL}{\log_2 n},$$

where n is the number of nodes in the network. The $\log_2 n$ term returns the number of symbols (bits) needed to code each node of the network taken separately, i.e. using a uniform code, in which all code words are of equal length. Since CL is the code length returned by Infomap, i.e. the number of symbols needed to code each node of the network given the community partition (that tries to exploit community information to use shorter code words), their ratio is telling us how much better is CL over the baseline. If $CL_{adj} \geq 1$, then the community division is using the same number of symbols (or more) than the ones needed without the community, otherwise the compression is effective, and the lower value the better partition. For this reason, CL_{adj} is scale independent.

The resulting plot of the CL_{adj} for all the networks generated is depicted in Fig. 15. As we can see, the adjusted code length decreases while approaching a cell size in the interval 5–10,000 m, that is our minimum, and then increases again. At cell size 8,000 m, the adjusted code length is slightly lower than 0.53, intuitively it means that the obtained code length is long as 53 % of the baseline. This confirms the topology analysis of the networks performed at the beginning of this section, that identified the most promising cell sizes at values smaller than 10,000 m. Moreover, the comparison

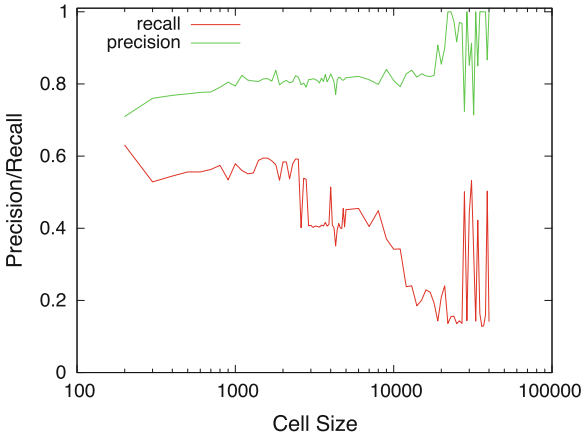


Fig. 14 The measures of precision and recall compared with the division of the territory into provinces

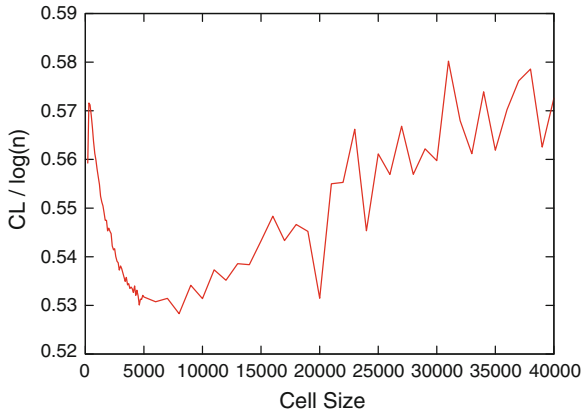


Fig. 15 The adjusted codelength values of the extracted networks

of the plots in Figs. 15 and 14 show that the communities discovered for grid sizes up to 2,000 m have comparable results at the cost of a complexity that decreases when the cell grid size increases. Beyond the grid size limit of 7–10,000 m the spatial grid is no more able to capture local mobility behavior and the corresponding communities and their complexity start getting worse.

6 Conclusion

In this chapter we explored the influence of the temporal and spatial dimension for the analysis of complex networks extracted from mobility data. We considered a large dataset of GPS trajectories, with a very precise temporal and spatial resolution. From

these trajectories, we derived different network perspectives: the first set is generated by defining time intervals (i.e. weekdays and weekends), the second set is generated by defining a set of multi-resolution spatial grids. We studied several network statistics over the extracted networks and we applied a community discovery algorithm to understand how the temporal and the spatial dimension affect the problem of detecting the actual borders of human mobility. The extensive experiments show that the choice of the appropriate temporally bounded data and spatial resolution of the grid is critical for the network study of mobility data. Temporally, data from periods of increased unpredictability can introduce noise and lower the quality of mobility prediction. Spatially, finer resolutions create over detailed networks where smaller components are associate to several small clusters. Large cell sizes, on the contrary, generate an excessive aggregation of local movements. We provided a framework to understand how to detect the optimal spatiotemporal tradeoff. We detected the optimal spatial resolution, that allows the correct generalization of local trips, that represent the majority of human mobility, and the reduction of model complexity of the extracted communities, which yield a compact code representation. We also detected that to maximize the usefulness of the mobility clusters, one has to rely on systematic trajectories, that are more frequent and predictable.

Acknowledgments The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n270833. We also acknowledge Octo Telematics S.p.A. for providing the dataset.

References

1. Andrienko GL, Andrienko NV, Bak P, Keim DA, Kisilevich S, Wrobel S (2011) A conceptual framework and taxonomy of techniques for analyzing movement. J Vis Lang Comput 22(3):213–232
2. Ankerst M, Breunig MM, Kriegel H-P, Sander J (1999) Optics: ordering points to identify the clustering structure. SIGMOD Rec 28(2):49–60. [Online] Available: http://doi.acm.org/10.1145/304181.304187
3. Bonchi F, Lakshmanan LVS, Wang WH (2011) Trajectory anonymity in publishing personal mobility data. SIGKDD Explor 13(1):30–42
4. Cho E, Myers SA, Leskovec J (2011) Friendship and mobility: user movement in location-based social networks. In: KDD
5. Coscia M, Giannotti F, Pedreschi D (2011) A classification for community discovery methods in complex networks. Stat Anal Data Min 4(5):512–546
6. Coscia M, Rinzivillo S, Giannotti F, Pedreschi D (2012) Optimal spatial resolution for the analysis of human mobility. In: ASONAM
7. Coscia M, Rossetti G, Giannotti F, Pedreschi D (2012) Demon: a local-first discovery method for overlapping communities. In: KDD, pp 615–623
8. Fortunato S, Lancichinetti A (2009) Community detection algorithms: a comparative analysis: invited presentation, extended abstract, ser. VALUETOOLS '09. ICST, pp 27:1–27:2
9. Jawad A, Kersting K, Andrienko NV (2011) Where traffic meets dna: mobility mining using biological sequence analysis revisited. In: GIS, pp 357–360
10. Krempl G, Siddiqui ZF, Spiliopoulou M (2011) Online clustering of high-dimensional trajectories under concept drift. In: ECML/PKDD (2), pp 261–276

11. Liu Y, Chen L, Pei J, Chen Q and Zhao Y (2007) Mining frequent trajectory patterns for activity monitoring using radio frequency tag arrays. In: PerCom, pp 37–46
12. Monreale A, Andrienko GL, Andrienko NV, Giannotti F, Pedreschi D, Rinzivillo S, Wrobel S (2010) Movement data anonymity through generalization. Trans Data Priv 3(2):91–121
13. Monreale A, Pinelli F, Trasarti R, Giannotti F (2009) WhereNext: a location predictor on trajectory pattern mining. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ser. KDD '09. ACM, New York, NY, USA, pp 637–646. [Online] Available: http://dx.doi.org/10.1145/1557019.1557091
14. Newman MEJ (2003) The structure and function of complex networks. SIAM Rev 45(2):167–256
15. Papadimitriou S, Sun J, Faloutsos C, Yu PS (2008) Hierarchical, parameter-free community discovery. In: ECML PKDD. Springer-Verlag, Berlin, Heidelberg, pp 170–187
16. Ratti C, Sobolevsky S, Calabrese F, Andris C, Reades J, Martino M, Claxton R, Strogatz SH (2010) Redrawing the map of great britain from a network of human interactions. PLoS ONE 15(12):e14248+
17. Rinzivillo S, Mainardi S, Pezzoni F, Coscia M, Pedreschi D, Giannotti F (2012) Discovering the geographical borders of human mobility. In: Kunstliche intelligenz (in press)
18. Rosvall M, Bergstrom CT (2011) Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. PLoS ONE 6(4):e18209
19. Sun Y, Han J, Aggarwal CC, Chawla NV (2012) When will it happen?: Relationship prediction in heterogeneous information networks. In: WSDM, pp 663–672
20. Szell M, Sinatra R, Petri G, Thurner S, Latora V (2011) Understanding mobility in a social petri dish. ArXiv e-prints, Dec 2011
21. Tang L, Liu H (2009) Relational learning via latent social dimensions. In: KDD. ACM, New York, NY, USA, pp 817–826
22. Thiemann C, Theis F, Grady D, Brune R, Brockmann D (2010) The structure of borders in a small world. PloS one 5(11):e15422+
23. Trasarti R, Pinelli F, Nanni M, Giannotti F (2011) Mining mobility user profiles for car pooling. In: KDD, pp 1190–1198
24. Verhein F, Chawla S (2008) Mining spatio-temporal patterns in object mobility databases. Data Min Knowl Discov 16(1):5–38. [Online] Available: http://dx.doi.org/10.1007/s10618-007-0079-5
25. Wang D, Pedreschi D, Song C, Giannotti F, Barabasi A-L (2011) Human mobility, social ties, and link prediction. In: KDD. ACM, New York, NY, USA, pp 1100–1108
26. Watts DJ and Strogatz SH (1998) Collective dynamics of 'small-world' networks. Nature 393(6684):440–442. [Online] Available: http://dx.doi.org/10.1038/30918

An Ant Based Particle Swarm Optimization Algorithm for Maximum Clique Problem in Social Networks

Mohammad Soleimani-pouri, Alireza Rezvanian and Mohammad Reza Meybodi

Abstract In recent years, social network services provide a suitable platform for analyzing the activity of users in social networks. In online social networks, interaction between users plays a key role in social network analysis. One of the important types of social structure is a full connected relation between some users, which known as clique structure. Therefore finding a maximum clique is essential for analysis of certain groups and communities in social networks. This paper proposed a new hybrid method using ant colony optimization algorithm and particle swarm optimization algorithm for finding a maximum clique in social networks. In the proposed method, it is improved process of pheromone update by particle swarm optimization in order to attain better results. Simulation results on popular standard social network benchmarks in comparison standard ant colony optimization algorithm are shown a relative enhancement of proposed algorithm.

Keywords Social network analysis · Clique problem · ACO · PSO

1 Introduction

Today online social networks are formed a new type of life due to some facilities and services for a wide ranges of ages such as young to old. Besides, there is no doubt about either influence or growth of social networks. Therefore, several

M. Soleimani-pouri (✉)
Department of Electrical, Computer and Biomedical Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran
e-mail: m.soleimani@qiau.ac.ir

A. Rezvanian · M. R. Meybodi
Soft Computing Laboratory, Computer and IT Engineering Department, Amirkabir University of Technology, Tehran, Iran
e-mail: a.rezvanian@aut.ac.ir

M. R. Meybodi
e-mail: mmeybodi@aut.ac.ir

F. Can et al. (eds.), *State of the Art Applications of Social Network Analysis*, Lecture Notes in Social Networks, DOI: 10.1007/978-3-319-05912-9_14,

many of researchers are focused on social network analysis aspects. It seems to be useful, studying certain structure of relation between users in social networks. One of the important user group structure associated with a full connected of some users, which known as clique structure [1, 2]. Several applications of finding clique are reported by researchers such as social networks analysis [3, 4], online shopping recommendation [5], evolution of social networks [6], scheduling [7], biochemistry [8], computer vision [9] and wireless sensor networks [10]. In literature, finding clique is categorized as NP-complete problems in graph theory [11].

In maximum clique problem, finding the largest complete subgraph is considered. It was introduced by Karp [12]. Various types of algorithms have been presented to solve clique problem, while evolutionary algorithms such as genetic algorithm (GA) and ant colony optimization (ACO) have been used more than others. Popular algorithm named *Ant-clique* algorithm, which make a maximal clique using sequential greedy heuristics based on ACO by adding vertices to the partial cliques iteratively [13]. Besides, another ACO based method hybridized by simulated annealing (SA) [14] and tabu search [15]. Although new hybrid algorithm obtained good results, they have a high complexity in practice.

In this study, Particle Swarm Optimization (PSO) algorithm has been applied as the heuristic to enhance the performance of ACO algorithm for finding the maximum clique in social network graph. Simulation results on social network benchmark are shown the better results in comparison with standard ACO algorithm. In the rest of this paper, Sects. 2 and 3 are consisted of ACO and PSO introduction respectively, in Sect. 4, proposed method is discussed. Simulation results on social networks datasets are reported in Sect. 5.

2 Ant Colony Optimization

Ant Colony optimization (ACO) algorithm works well for solving several discrete problems. The basic algorithm of ACO was proposed by Dorigo as a multi agent approach in order to solve traveling salesman problem (TSP) [16]. The main idea of ACO algorithm is inspired from the behavior of seeking out food by colonies of ants. Ants search their environment randomly to find food. They return some of the food to their nest once found a food and leave pheromone in their return path. The amount of pheromone left on their path depends on quality and size of the food source and it evaporates gradually. Remaining pheromones will persuade other ants to follow the path and just after a short time, majority of the ants will trace the shorter path which is marked with stronger pheromone. Procedure of ACO algorithm has been presented in Fig. 1.

During running of the algorithm, ants first produce different solutions randomly in the main loop after initialization. Afterwards, the solutions are improved by updating the pheromones and using a local search optionally. According to the problem and graph traverse, pheromones set on vertices or edges of graph. Traversing the edge

Fig. 1 Pseudo-code of ACO algorithm [17]

Algorithm 1 ACO algorithm

```
procedure ACO_MetaHeuristic
    while (termination_conditions)
        generateSolutions()
        daemonActions() {Optional}
        pheromoneUpdate()
    endwhile
end procedure
```

between vertices i and j depends on the probability of edge which is calculated as below:

$$p_{ij}^{k} = \frac{\left(\tau_{ij}^{\alpha}\right)}{\sum\left(\tau_{ij}^{\alpha}\right)} \tag{1}$$

where, ${p_{ij}}^{k}$ is probability of traversing the edge between vertices i and j, while ${\tau_{ij}}^{\alpha}$ is amount of pheromone present on the above mentioned edge. An optional local search can contribute to improvement of the results prior to updating the pheromones. However, method of updating the pheromones can be like this:

$$\tau_{ij}^{t+1} = (1-\rho)\,\tau_{ij}^{t} + \Delta\tau_{ij}^{t} \tag{2}$$

where, ρ is evaporation rate of pheromone, ${\tau_{ij}}^{t+1}$ is amount of new pheromone for edge between i and j, ${\tau_{ij}}^{t}$ is amount of current pheromone for edge between i and j, ${\Delta\tau_{ij}}^{t}$ is amount of reinforced pheromone for proper solutions which can be calculated from the following equation.

$$\Delta\tau_{ij}^{t} = \begin{cases} 1 & \text{if } \tau_{ij}^{t} \in \text{good solution} \\ 0 & \text{Otherwise} \end{cases} \tag{3}$$

3 Particle Swarm Optimization

Particle swarm optimization (PSO) is a population based optimization technique developed by Eberhart and Kennedy in 1995, which inspired from social behavior of birds seeking for food. In this method, group of birds seek for food in a specified space randomly. Birds follow the bird with the shortest distance to food in order to find position of the food [18].

Every solution in PSO algorithm which is called a particle corresponds to a bird in their pattern of social movement. Each particle has a value of fitness calculated by a fitness function. A particle bears greater fitness once it is located in a closer position

to the target (food in the model of moving birds) within search space. Moreover, any particle represents a speed which is responsible to direct the particle. A particle will keep going through the problem space by following the optimum particle at current state.

A group of particles (solutions) are created randomly at the beginning of particle swarm optimization algorithm and they try to find the optimum solution through being updated among generations. A particle is updated in each step using the best local and global solution. The best position a particle has ever succeeded to reach is called *pbest* and saved while the best position achieved by the population of particles is named *gbest*. Velocity and location of each particle will be updated using Eqs. (4) and (5) in each step of implementing the algorithm after finding the best local and global values.

$$v_i^{t+1} = wv_i^t + c_1 r_1 \left(pbest_i^t - x_i^t\right) + c_2 r_2 \left(gbest_i^t - x_i^t\right) \tag{4}$$

$$x_i^{t+1} = x_i^t + v_i^t \qquad i = 1, \ldots, m \tag{5}$$

where, v_i is the velocity of ith particle and x_i is the current position of it. r_1 and r_2 are random values in the range of (0,1). c_1 and c_2 are learning parameters usually assumed equal ($c_1 = c_2$). w is inertia weight which is considered as a constant or variable coefficient as random, linear, nonlinear and adaptive [19]. PSO has been used in various applications [20–22] and this research utilizes it to improve the amount of pheromones.

4 Proposed Algorithm

High complexity was a major drawback of the previous heuristic methods for solving the clique problem since it significantly adds to the volume of calculations. All methods provided so far apply the following relation to calculate $\Delta\tau$ although the proposed algorithm take the advantage of PSO algorithm to improve results and reduce complexity.

$$\Delta\tau^{t+1} = \frac{1}{(1 + |G\text{-best}| - |\text{best-tour}|)} \tag{6}$$

This algorithm has used the hybrid of ACO and PSO algorithms in order to find the maximum clique in a graph. For this purpose, some ants are placed initially on the graph and follow paths to find the maximum clique. After evaporation of existing pheromones, proper path is updated by its amount on the edges using PSO algorithm. This procedure is repeated until the optimum clique is obtained on the desired graph. Determining the amount of pheromone through PSO is such that the total pheromone measured at this step and the total pheromone associated with the best answer up

Algorithm 2 proposed Algorithm

For each ant choose randomly a first vertex $v_f \in V$
$C \leftarrow \{v_f\}$
candidates$\leftarrow \{v_f/(v_i,v_i) \in E\}$
while candidates$\neq \emptyset$ do
    choose a vertex $v_i \in$ candidate with probability by equation (1)
    $C \leftarrow C \cup \{v_i\}$
    candidates$\leftarrow$candidates$\cap \{v_f/(v_i,v_f) \in E\}$
endwhile
evaporate()
Evaluation best current clique for all ant
Pheromone update $\Delta\tau$ based on PSO by equation (7)

Fig. 2 Pseudo-code of proposed algorithm based on ACO and PSO

to present step will be calculated taking into account the amount of pheromones at current step. Now, $\Delta\tau$ for the reinforced pheromone of the desired clique will be calculated with PSO algorithm using the following equation:

$$\Delta\tau^{t+1} = \Delta\tau^{t} + V^{t} \tag{7}$$

where, ${\Delta\tau_{ij}}^{t}$ is amount of reinforcement for current pheromone and ${\Delta\tau_{ij}}^{t+1}$ is amount of reinforcement for new pheromone, while V^t gives the amount of change which can be achieved from this equation:

$$V^{t+1} = c_1 r_1 (p\tau - \Delta\tau) + c_2 r_2 (g\tau - \Delta\tau) + c_3 V^{t} \tag{8}$$

where, V^{t+1} is the new value of V^t. r_1 and r_2 are two random values within range of (0,1), while c_1, c_2 and c_3 are learning parameters ($c_1 = c_2 = c_3$). $p\tau$ and $g\tau$ are considered as the pheromone correspondent with the best current clique and the best clique found so far, respectively. In this case, discovering the changed amount of pheromone will be implemented much more intelligent.

Taking into consideration the mentioned items, general routine of proposed the algorithm can be presented in Fig. 2.

5 Simulation Results

5.1 Comparison of the Proposed Algorithm

For evaluation of the proposed method, experiments applied on some popular social network datasets [19] and DIMACS graphs [2]. The descriptions of these networks are listed in the Table 1.

Table 1 Descriptions of social network datasets and DIMACS graphs for experiments

Name	Descriptions	Nodes	Edges
SN_I	Zachary's karate club	34	78
SN_II	Common adjective and nouns in "David Copperfield"	112	425
SN_III	Neural network of the nematode C. Elegans	297	8479
SN_IV	Social network of dolphins, Doubtful Sound, New Zealand	62	159
SN_V	Pajek network: Erdos collaboration network 971	472	1314
SN_VI	Pajek network: Erdos collaboration network 991	492	1417
SN_VII	Pajek network: World Soccer, Paris 1998	35	295
SN_VIII	Pajek network: graph and digraph glossary	72	118
SN_IX	Pajek network: Slovenian journals 1999–2000	124	823168
SN_X	Co-authoship of scientists in network theory and experiments	1589	1190
SN_XI	Pajek network: SmaGri citation network	1059	4919
SN_XII	Email interchange network, Univ. of Rovira i Virgili,Tarragona	1133	5451
DC_I	DIMACS graphs: C250.9	250	27984
DC_II	DIMACS graphs: gen200_p0.9_44	200	17910
DC_III	DIMACS graphs: keller4	171	9435
DC_IV	DIMACS graphs: MANN_a27	378	70551
DC_V	DIMACS graphs: p_hat700-2	700	121728
DC_VI	DIMACS graphs: brock400_2	400	59786
DC_VII	DIMACS graphs: C1000_9	1000	450079
DC_VIII	DIMACS graphs: hamming10_4	125	6963

Topologies of some social networks and DIMACS are presented in Fig. 3 as the most popular of the social network datasets.

The setting of parameters for experiment is listed in below. It is noted that choosing different values for improving the results is also possible.

In this chapter has used 30 ants with $\rho = 0.95, \Phi = 0.0002, \Delta\tau_{initial} = 0, \tau_{\min} = 0.01, \tau_{\max} = 6$.

Meanwhile, parameters of PSO were initialized as $V = 0$, $c_1 = c_3 = 0.3$, and $c_2 = 1 - c_1$. α and ρ were given the following values based on the experiments done and t is the number of iterations.

$$\alpha(t) = \begin{cases} 1 & t \leq 100 \\ 2 & 100 < t \leq 400 \\ 3 & 400 < t \leq 800 \\ 4 & t > 800 \end{cases} \tag{9}$$

$$\rho(\text{t+1}) = \begin{cases} (1-\varphi)\rho(\text{t}) \\ 0.95 \qquad \text{if } \rho(\text{t}) > 0.95 \end{cases} \tag{10}$$

Average of 10 independent runs with 1,000 iterations for each Social network dataset implementation have been listed in Table 2, and 30 independent runs with 1,500 iteration for DIMACS have been presented in Table 3 for proposed method

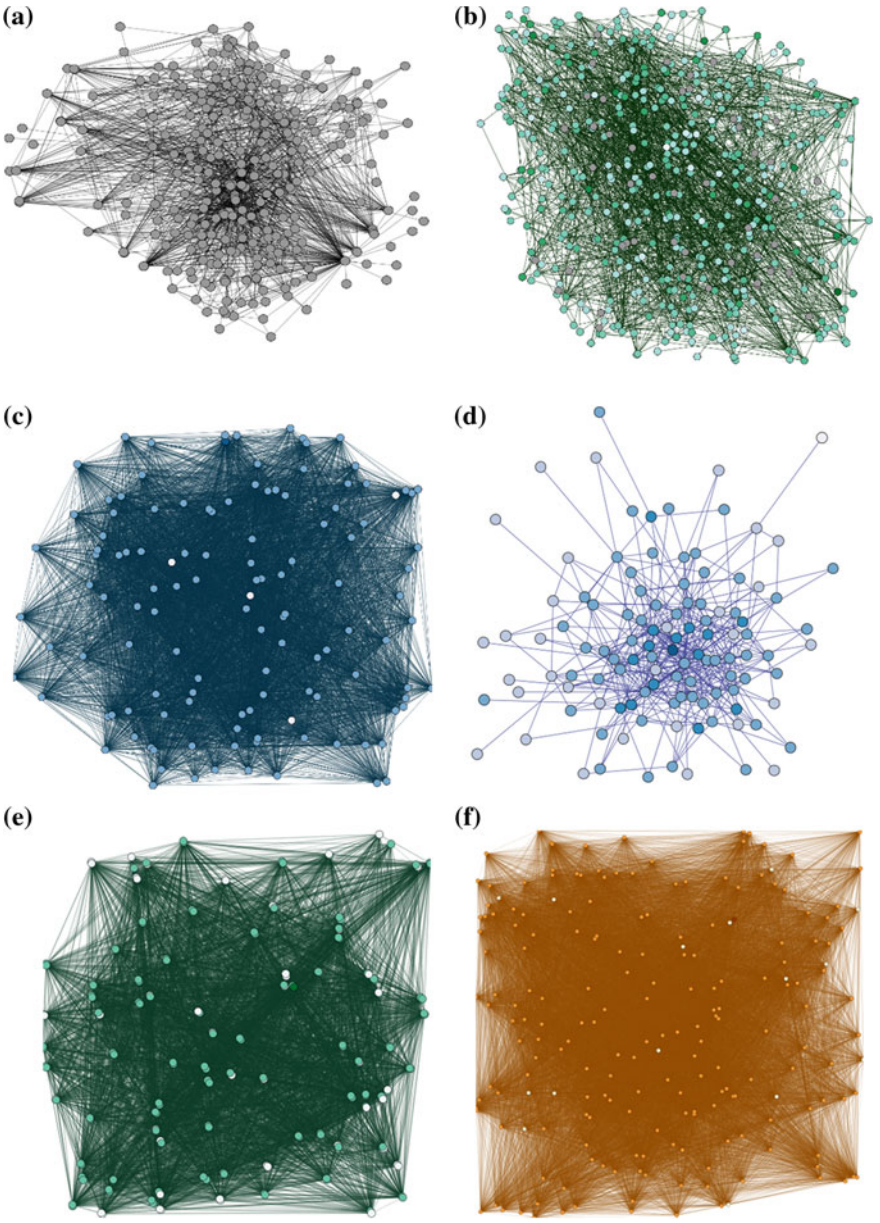


Fig. 3 Visulization of some datasets for experiments. **a** SN_III. **b** SN_VI. **c** SN_IX. **d** SN_II. **e** DC_III. **f** DC_II

Table 2 Simulation results for finding a maximum clique in social network datasets

Graph	ACO				ACO-PSO			
	Best	Avg	Std	Run-time	Best	Avg	Std	Run-time
SN_I	5	4.991	0.080	64.03	5	4.995	0.070	11.84
SN_II	5	4.709	0.495	355.81	5	4.783	0.412	23.37
SN_III	7	5.521	1.568	1121.71	7	5.543	1.482	49.47
SN_IV	5	4.991	0.094	132.12	5	4.998	0.044	15.69
SN_V	7	8.753	0.861	5853.59	7	5.848	0.667	105.11
SN_VI	7	5.961	0.924	6281.35	7	6.011	0.819	111.89
SN_VII	5	3.910	0.293	55.15	5	4.118	0.322	10.742
SN_VIII	4	3.943	0.232	153.66	4	3.985	0.121	12.94
SN_IX	4	3.017	0.316	438.87	4	3.185	0.299	22.07
SN_X	3	2.175	0.434	2142.03	3	2.253	0.371	118.59
SN_XI	8	6.025	0.815	9153.65	8	6.409	0.765	481.34
SN_XII	12	7.562	2.370	11109.17	12	7.997	2.240	529.04

Table 3 Simulation results for finding a maximum clique in DIMACS datasets

Graph	ACO				ACO-PSO			
	Best	Avg	Std	Run-time	Best	Avg	Std	Run-time
DC_I	44	43.20	0.761	43343.98	44	43.83	0.461	27978.04
DC_II	44	41.50	2.270	23992.76	44	42.90	1.863	19485.80
DC_III	11	11.00	0.000	8294.87	11	11.00	0.000	8417.40
DC_IV	126	125.60	0.498	194254.88	126	126.00	0.000	200678.54
DC_V	44	43.60	0.621	89776.41	44	44.00	0.000	89539.63
DC_VI	25	23.53	0.628	24730.51	29	24.30	1.087	29580.20
DC_VII	63	59.86	1.569	265326.84	67	64.06	1.172	276632.40
DC_VIII	40	36.30	1.556	151290.21	40	38.60	1.404	177169.91

(ACO-PSO) and ACO algorithm respectively, including the maximum (Best), average (Avg), standard deviation (Std) and run-time of algorithm for finding a maximum clique in each graph datasets.

Tables 2 and 3 indicate that the proposed method (ACO-PSO) produces better results in comparison with ACO method since the proposed approach is an appropriate method to update pheromones of the traversed paths for ants in calculating the optimum clique.

5.2 Comparison of Convergence of the Proposed Algorithm

Convergence behavior of the proposed algorithm in comparison standard ACO have been presented along running for some graphs. Figure 4 shows an average of 30

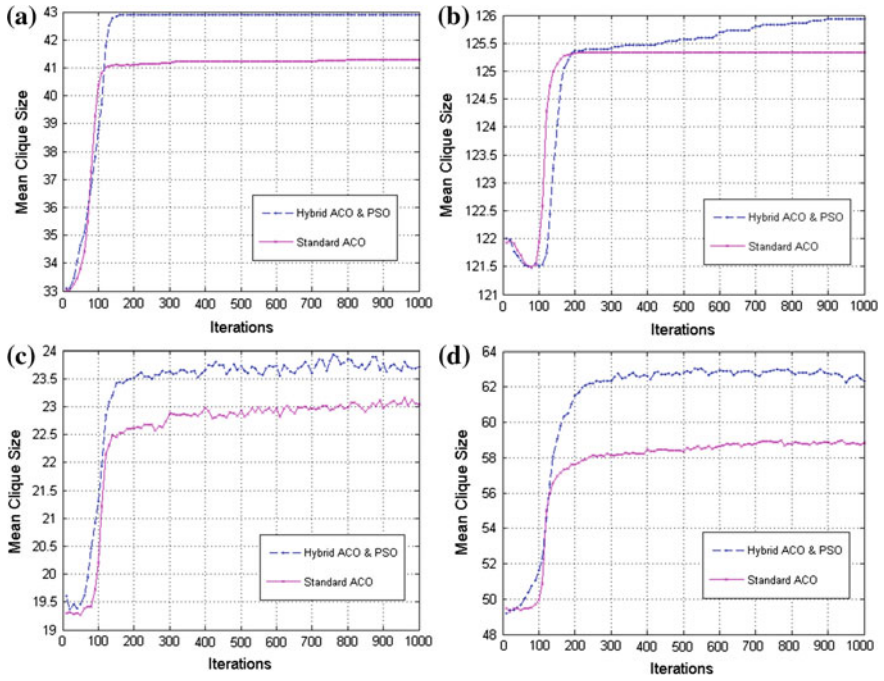


Fig. 4 Convergence behavior of the proposed algorithm in comparison standard ACO. **a** DC_II. **b** DC_IV. **c** DC_VI. **d** DC_VII

independent runs for 1,000 periods within intervals of 10 periods for some network graphs listed in table 1.

In these figures can be observed that proposed algorithm has excellent result compared with standard ACO.

6 Conclusion

A new hybrid algorithm has been presented in this paper using ACO and PSO (ACO-PSO) for finding the maximum clique problem. Traditional algorithms suffered high complexity while the hybrid proposed algorithm just change the process of update pheromone. It has been shown that the new algorithm was able to improve the basic ACO algorithm as simply and quickly. Simulation results on popular social network datasets indicate the suitable results for proposed algorithm in comparison with the ACO algorithm.

References

1. Easley D, Kleinberg J (2010) Networks, crowds, and markets: reasoning about a highly connected world. Cambridge University Press, Cambridge
2. Soleimani-Pouri M, Rezvanian A, Meybodi MR (2012) Solving maximum clique problem in stochastic graphs using learning automata. In: 2012 Fourth international conference on computational aspects of social networks (CASoN), pp 115–119
3. Sadi S, Öğüdücü S, Uyar A (2010) An efficient community detection method using parallel clique-finding ants. In: 2010 IEEE congress on evolutionary computation (CEC), pp 1–7
4. Fadigas IS, Pereira HBB (2013) A network approach based on cliques. Physica A 392(10):2576–2587
5. Yang Q, Zhou P, Zhang H, Zhang J (2011) Clique discovery based on user similarity for online shopping recommendation. Inf Technol J 10:1587–1593
6. Yan F, Cai S, Zhang M, Liu G, Deng Z (2012) A clique-superposition model for social networks. Science China Information Sciences, pp 1–19
7. Liua Y, Zhanga D, China FYL (2011) A clique-based algorithm for constructing feasible timetables. Optim Methods Softw 26(2):281–294
8. Butenko S, Wilhelm WE (2006) Clique-detection models in computational biochemistry and genomics. Eur J Oper Res 173(1):1–17
9. Liu J, Lee YT (2001) Graph-based method for face identification from a single 2D line drawing. IEEE Trans Pattern Anal Mach Intell 23(10):1106–1119
10. Wang L, Wei R, Lin Y, Wang B (2010) A clique base node scheduling method for wireless sensor networks. J Netw Comput Appl 33(4):383–396
11. Kleinberg J, Tardos E (2005) Algorithm design. Addison Wesley
12. Karp RM (1972) Reducibility among combinatorial problems. Complex Comput Comput 40(4):85–103
13. Fenet S, Solnon C (2003) Searching for maximum cliques with ant colony optimization. In: Cagnoni S, Johnson CG, Cardalda JJR, Marchiori E, Corne DW, Meyer J, Gottlieb J, Middendorf M, Guillot G, Raidl R, Hart E (eds) EvoWorkshops. LNCS, vol 2611. Springer, Heidelberg, pp 236–245
14. Xu X, Ma J, Lei J (2007) An improved ant colony optimization for the maximum clique problem. In: 3rd International conference on natural computation (ICNC), pp 766–770
15. Al-Fayoumi M, Banerjee S, Mahanti PK (2009) Analysis of social network using clever ant colony metaphor. World Acad Sci Eng Technol 53:970–974
16. Bonabeau E, Dorigo M, Theraulaz G (1999) Swarm intelligence. Oxford University Press, Oxford
17. Socha K, Dorigo M (2008) Ant colony optimization for continuous domains. Eur J Oper Res 185(3):1155–1173
18. Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: IEEE international conference on neural networks, pp 1942–1948
19. Amiri F, Yazdani N, Faili H, Rezvanian A (2012) A novel community detection algorithm for privacy preservation in social networks. In: Abraham A, Thampi SM (eds) ISI 2012. AISC, vol 182. Springer, Heidelberg, pp 443–450
20. Nabizadeh S, Faez K, Tavassoli S, Rezvanian A (2010) A novel method for multi-level image thresholding using particle swarm Optimization algorithms. In: 2010 2nd International conference on computer engineering and technology (ICCET), pp v4-271–275
21. Nabizadeh S, Rezvanian A, Meybodi MR (2012) A multi-swarm cellular PSO based on clonal selection algorithm in dynamic environments. In: 2012 International conference on informatics, electronics and vision (ICIEV), pp 482–486
22. Nabizadeh S, Rezvanian A, Meybodi MR (2012) Tracking extrema in dynamic environment using multi-swarm cellular pso with local search. Int J Electron Inform 1(1):29–37

XEngine: An XML Search Engine for Social Groups

Kamal Taha

Abstract We introduce in this chapter an XML user-based Collaborative Filtering system called XEngine. The framework of XEngine categorizes social groups based on ethnic, cultural, religious, demographic, or other characteristics. XEngine outputs ranked lists of content items, taking into account not only the initial preferences of the user, but also the preferences of the various social groups, to which the user belongs. The user's social groups are inferred *implicitly* by the system without involving the user. XEngine constructs the social groups and identifies their preferences *dynamically* on the fly. These preferences are determined from the preferences of the social groups' member users using a group modeling strategy. XEngine can be used for various practical applications, such as Internet or other businesses that market preference-driven products. We experimentally compared XEngine with three existing systems. Results showed marked improvement.

1 Introduction

With the growth of massive information on the Web, it has become increasingly difficult to search for useful information. As one of the most promising approaches to alleviate this overload, recommender systems have emerged in domains such as E-commerce and digital libraries. In general, recommendation systems suggest items or products, by analyzing what users with similar tastes have chosen in the past. One of the successful recommendation tools is Collaborative Filtering (e.g., [12]). Collaborative Filtering (CF) is the process of filtering for information using the opinion of other people. It aims at predicting the user interest for a given item based on a collection of profiles of users who have similar interests. The underlying

K. Taha (✉)
Department of Electrical and Computer Engineering, Khalifa University of Science, Technology, and Research, Abu Dhabi, United Arab Emirates
e-mail: kamal.taha@kustar.ac.ae

F. Can et al. (eds.), *State of the Art Applications of Social Network Analysis*, Lecture Notes in Social Networks, DOI: 10.1007/978-3-319-05912-9_15,

assumption of CF approach is that those who agreed in the past tend to agree again in the future.

A number of CF algorithms have been proposed. There are two major classes of these algorithms [5], memory-based and model-based approaches. Memory-based CF (e.g., [5]) predicts a user's preference based on his/her similarity to other users in the database. Model-based CF first learns a descriptive model of the user preferences and then uses it for providing item recommendation. The advantage of the memory-based methods over their model-based alternatives is that less parameters have to be tuned. Existing memory-based CF methods, mainly user-based (e.g., [5]) and item-based (e.g., [7]) methods, predict new ratings by aggregating rating information from either similar users or items. Given an unknown test rating to be estimated, user-based CF measures similarities between test user and other users. Item-based CF measures similarities between test item and other items.

We introduce in this chapter an XML [6] user-based CF system called XEngine. The framework of XEngine categorizes social groups based on ethnic, cultural, religious, demographic, age, or other characteristics. For example, people of ethnic group E_X; people who follow religion R_Y; and people who live in neighborhood N_Y can all be considered to form various social groups. XEngine outputs ranked lists of content items, taking into account not only the initial preferences of the active user, but also the *preferences of the active user's various social groups*. Consider for example a Mexican-American user. The user belongs to social groups Mexicans and Americans: the portion of Mexicans living in the USA. The results of a query submitted by this user will be *filtered* and *ranked* based on the union of the interests of social groups "Mexicans" and "Americans". The social groups to which a user belongs usually have *class-subclass* relationships. A subclass social group has its own properties while inheriting the properties of its superclass(es).

In the framework of XEngine, user characteristics (e.g., social groups) are inferred *implicitly* by the system without involving the user. That is, the user is not required to reveal the social groups to which he belongs. The user is determined whether or not he/she belongs to a social group G by matching his/her rating pattern with the rating pattern of G. XEngine constructs social groups and also identifies their preferences *dynamically* on the fly. We developed formal concepts and algorithms that identify the preferences of various social groups dynamically on the fly. These preferences are determined from the preferences of the social groups' member users using a group modeling strategy. XEngine can be used for various practical applications, such as Internet or other businesses that market preference-driven products. *We experimentally compared XEngine with three existing systems*. Results showed marked improvement. *A demo of XEngine is available at*: http://dbse1.uta.edu/kamal/XEngine/.

The rest of this chapter is organized as follows. In Sect. 2, we present related work. In Sect. 3, we outline our approach. In Sect. 4, we describe how the ratings of a social group can be initialized dynamically from Web pages publishing information about the social group. In Sect. 5, we describe how the social group to which the user belongs can be identified implicitly by the system. In Sect. 6, we describe how we model social groups graphically. In Sect. 7, we describe how results are filtered

and ranked based on the ratings of the user's social group. We present the system architecture in Sect. 8. We present our experimental results in Sect. 9 and conclusions in Sect. 10.

2 Related Work

There have been a number of researches in filtering based on group profiling [1, 11, 18, 21, 24, 27]. In most of these works, a group is formed based on common interests of its members on an item(s)/features.

O'Connor et al. [18] describes how a combination of collaborative and demographic filtering can be used to recommend product bundles. It describes how stored data is used to recommend a combination of tourist services. Gomes and Canuto [11] presents Caracará, a system for searching and mining information on the World Wide Web, using a dynamic grouping process. Carcará groups Internet users according to their profile. After that, the system makes suggestions of URLs which are likely to be useful for the users of these groups. Aimeur et al. [1] creates categories of users having similar demographic characteristics, and tracks the aggregate buying behavior of users within these categories. Recommendations for a new user are issued by applying the aggregate buying preferences of previous users in the category to which the user belongs. Wang et al. [27] presents a model for supporting social groups in an Ubicomp environment. There must be consensus between group members in order for a person to be a member of the group.

Symeonidis et al. [21] proposes an approach/system called Feature-Weighted Nearest Biclusters (FWNB), which is the closest and most relevant to our proposed work. In Sect. 9, we experimentally compare **FWNB** with XEngine. The following is an overview of FWNB:

An overview of FWNB [21]: The paper proposes to capture the interaction between users and their favorite features by constructing feature profile for users. Users are grouped into biclusters (i.e., *group of users which exhibit highly correlated ratings on groups of items*). Each bicluster acts like a community for its corresponding items. Each bicluster's item features are weighted. The weighted value of feature f for user u $W(u, f)$ is calculated as follows: $W(u, f) = \mathrm{FF}(u, f) * \mathrm{IUF}(f)$, where $\mathrm{FF}(u, f)$ is the number of times f occurs in the profile of user u; and, $\mathrm{IUF}(f)$ is the *inverse user frequency* and is calculated as: $IUF(f) = \log(|U|/UF(f))$, where $|U|$ is the *total number of users;* $\mathrm{UF}(f)$ is the *number of users in which feature f occurs at least once*.

2.1 The Following are Outlines of the Novelty of Our Approach Compared to Existing Approaches

1. Employing context-driven group profiling techniques:
   Most current user-based CF algorithms suffer the following limitation. When they determine the users having similar rating pattern as the active user, the number

of these users is usually much less than the actual number. The reason is that most of these algorithms consider *only* item features rated by the active user and co-rated by other users, and users usually rate only a *subset* of an item's features. We observe that we can account for all users that have similar interests as the active user by employing a mechanism similar to *friend-of-a friend* ontology. For example, if: (1) user u_x has preference on feature f_i, (2) user u_y has preference on features f_i and f_j, and (3) user u_z has preference on feature f_j. By applying the *friend-of-a friend* ontology, we will find that *all* three users may have *common* interests. Moreover, these interests may include features other than f_i and f_j, if the three users belong to some common *context*. A context is a characteristic that define a group based on culture, ethnicity, religion, demography, or other characteristics. Therefore, we propose grouping users based on their contexts' profiles rather than solely on their rating patterns. If an active user u_x and another user u_y rated *different* features of an item, but both have similar rating pattern as a social group G, it is most likely they both have similar interests, which are the interests of the social group G. The interests of u_x can be predicted based on the interests of the social group G. Some of the member users of G may have rated features other than those rated by user u_y.

2. Employing multi-context group profiling techniques:
   The smaller a social group is, the more granular and specific its interests are. We propose smaller granularity of group classes formed from the intersections of different contexts. That is, a group from this class is composed of an aggregation of people sharing common *multi-context* profile. For example, the *portion* of ethnic group E_X who follow religion R_Y forms a multi-context group.
3. Coupling multi-context profiling with implicit and dynamic identification techniques:
   We constructed previously a system called SPGProfile [23], which also employs the concept of group profiling. However, SPGProfile uses *static* and *explicit* mechanisms as follows: (1) the user is required to reveal to the system *explicitly* the social groups, to which he belongs,[1] and (2) the preferences of a social group are determined *statically* from preference data acquired from subjects belonging to the social group (e.g., *via interviews*); these preferences will NOT be updated and optimized based on the preferences of new member users of the social group. *A link to SPGProfile demo is provided in the XEngine's Web site*.

2.2 Key Differences Between XEngine and SPGProfile

1. XEngine identifies the social groups to which the user belongs *implicitly*. That is, the user is not required to reveal to the system the social groups, to which he

[1] Collecting characteristics from users themselves can be *intrusive* into the user interaction. Moreover, users are usually unwilling to spend extra effort to explicitly specify their characteristics (e.g., *social groups*) [20]. Therefore, XEngine is *an improvement* over SPGProfile.

belongs. As for SPGProfile, the user is required to reveal to the system *explicitly* his/her social groups.
2. XEngine constructs social groups and identifies their preferences *dynamically* on the fly. The preferences/ratings of a social group are dynamically *updated* and *optimized* based on the ratings of *each* new member user belongs to the social group. As for SPGProfile, the system administrator needs first to identify subjects belonging to each social group (e.g., *via phone interviews*). He then provides them with a GUI to rate their preferences on features. These preferences act as the preferences of the social group and *will NOT be updated* based on the preferences of new member users belonging to the social group.
3. By crawling Web sites, XEngine *initializes* the preferences of social groups *dynamically* from Web pages that publish information about them. It also employs a strategy that rates these preferences. SPGProfile does not do so.

3 Concept Used in the Chapter and Outline of the Approach

3.1 Definition of Key Concepts

Notation 3.1 ***An item feature***: We use the term "feature" to refer to the name of a data element in an XML document. A feature reveals a distinctive and essential aspect of an item.

We use the terms *user preference* and *user interest* interchangeably throughout the chapter.

Definition 3.1 *User preference* (*interest*): A user preference W is an ordered set of some data dimensions D, i.e., $W = \{w_1, w_2, \ldots, w_m\}$, where $w_i \in D$. It follows the order such that $w_1 > w_2 > \cdots > w_m$ where $w_i > w_j$ represents that the user prefers w_i to w_j.

We use the term "domain" to mean an area of activity, belief, culture, ethnicity, demography, or the like. A *Single-Domain Group* (*SDG*) is a group of people sharing common *domain interests*. For example, each of ethnic group E_X and religious group R_Y represents a SDG.

Definition 3.2 *Single-Domain Group* (*SDG*): A SDG is an aggregation G of individual users, where for each $x, y \in G$ $(x \neq y)$: x and y share a common and distinctive culture, ethnicity, religion, demography, or the like. That is, x and y share the same interests of *only one* domain group.

The smaller a social group is, the more granular and specific its interests are. Therefore, we introduce another class of social groups called *Multi-Domain Group* (*MDG*) whose size is usually smaller than a SDG. A MDG is composed of an aggregation of people sharing common *multi-domain* interests. Thus, a MDG is formed from

the *intersection* of two or more SDGs. For example, the *portion* of ethnic group E_X who follow religion R_Y and live in neighborhood N_Y (i.e., the intersection of $E_X \cap R_Y \cap N_Y$) forms a MDG. The interests of a MDG are the *union* of the interests of the SDGs forming it. Thus, the interests of a MDG are *more specific* than the interests of each of the SDGs forming it. To fine-grain a user's query results, XEngine outputs a filtered and ranked list of items taking into account the preferences of the user's MDG.

Definition 3.3 ***Multi-Domain Group*** (***MDG***): Let S be the set of all SDGs that exist. A MDG is an aggregation G of individual users, where: $\forall x \in G{:}\, x$ shares the same interests of $\forall s \subseteq S{:}|s| \geq 2$. That is, G is formed from the intersection $\bigcap_{SDG_i \in s, s \subseteq S} SDG_i$.

We model the user's *initial preferences* as a set $D = \{(a_1, w_1), \ldots, (a_m, w_m)\}$, where: a_i denotes item feature i and w_i a weight on a_i. The weight w_i is a value scaled between 0 and 1. A complete set of features is presented to users to determine their relevance. The system provides the users with a *graphical user interface* (GUI) to reveal their *initial* preferences and *weights* on a feature. Based on the weights of a SDG's member users on features, each feature is given a score. This score reflects the importance of the feature to the SDG relative to other features. We adopt the following strategy for determining these scores:

> Each *feature* is assigned a *score*. This score is based on the difference between the *number of times* the feature *beats* other features (i.e., *assigned a higher weight by the members of the SDG*), and the number of times it *loses*.

Definition 3.4 ***A score of a feature for a SDG***: Let $a \succ b$ denote: the *number of times* the *members* of a SDG considered feature a as a preference is *greater* than that of feature b. Let $c(a)$ denote the score of feature a. Given the dominance relation $\succ$ on a set F of features rated by the SDG, the score $c(a)$ of feature "a" equals $|\{b \in F : a \succ b\}| - |\{b \in F : b \succ a\}|$.

The following are some of the characteristics of this scoring strategy: (1) the sum of the scores of all features is always zero, and (2) the *highest and lowest possible* scores are $(n - 1)$ and $-(n - 1)$ respectively, where n is the number of features. We normalize the scores by first adding the *absolute of the most negative score* to all scores and then normalizing the resulting values. We now introduce a running example to illustrate some of the concepts in this chapter.

Example 1 Consider Table 1, which shows the ratings of three SDG's members on 10 features (f_{1-10} are the features and u_{1-3} are the users). The rating scale is between [0–10]. Based on the ratings in Table 1, the "beats" and "looses" of each feature are shown in Table 2. The symbol "+" denotes that a feature beat a corresponding one (i.e., *rated higher by the majority of users*), while "−" denotes it lost. For example, feature f_1 beat feature f_7. A zero means: two features beat each other the same number of times and also lost to each other the same number of times. The raw before the last one in Table 2 shows the score of each feature computed using the strategy described in Definition 3.4. The last raw shows the normalized scores.

Table 1 Weighted user-feature matrix

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
u_1	4	9	3	2	8	2	4	6	10	7
u_2	3	8	4	5	6	4	2	2	8	6
u_3	5	10	7	6	7	7	5	3	7	5

Table 2 Beats/Looses, score, and normalized score of each feature based on the ratings in Table 1

Feature	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
f_1	0	+	+	+	+	+	−	−	+	+
f_2	−	0	−	−	−	−	−	−	0	−
f_3	−	+	0	−	+	−	−	−	+	+
f_4	−	+	+	0	+	0	−	−	+	+
f_5	−	+	−	−	0	−	−	−	+	−
f_6	−	+	+	0	+	0	−	−	+	+
f_7	+	+	+	+	+	+	0	0	+	+
f_8	+	+	+	+	+	+	0	0	+	+
f_9	−	0	−	−	−	−	−	−	0	−
f_{10}	−	+	−	−	+	−	−	−	+	0
Score	−5	+8	+1	−2	+5	−2	−8	−8	+8	+3
Normalized score	0.04	0.2	0.1	0.08	0.16	0.08	0	0	0.2	0.14

3.2 Outline of the Approach

1. Predefining and modeling SDGs:
   The system administrator predefines SDGs and their domains from published studies. SDGs are usually defined in publications such as published government censuses and statistical studies. After the system administrator inputs the SDGs' data to XEngine, the system models it as *ontology-driven* graphical representation. In this modeling technique, each SDG is represented as a vertex, and the relationships between SDGs are represented as edges. An edge depicts the ontological relationship between two SDGs. Section 6.1 describes this process in more details.
2. Initializing the preferences of a SDG:
   The preferences and ratings of a SDG are first initialized.[2] The ratings of a SDG are *initialized* from preference data concerning the SDG acquired from published scientific studies. The initialization is done either *dynamically* from Web pages or *statically* from published hard-copy studies. The system identifies the *first corpus of users* belonging to a SDG by matching their ratings with the initialized ratings of the SDG. Section 4 describes this process.

[2] Then, these ratings will be *updated and optimized* constantly based on the ratings of each member user belongs to the SDG (see Step 3).

3. Identifying the members of a SDG implicitly and updating the preferences of the SDG Dynamically:
   After the ratings of a SDG are initialized, they will be *updated* and *optimized* based on the ratings of the *member users* of the SDG. As *each* new user is identified by the system (*implicitly*) as belonging to the SDG, the ratings of the SDG will be *optimized* and *updated* (*dynamically*) based on the ratings of this user, using a *group modeling strategy*. The system identifies (*implicitly*) new member users of the SDG *by matching their ratings with the ratings of the SDG*. Section 5 describes this process.
4. Identifying the user's MDG and filtering results:
   In response to a user's query/rating, XEngine performs the following steps: (1) constructs an XQuery query [6] equivalent to the user's query, (2) identifies *all possible* MDGs that exist because of the interrelations between SDGs, and identifies *implicitly* the smallest MDG to which the user belongs, and (3) *filters and ranks* the user's query results based on the ratings of the user's smallest MDG. These steps are described in details in Sects. 6.2 and 7.

4 Initializing the Ratings of a SDG

The ratings of a SDG can be initialized either *statically* from hard-copy published studies (*see* Sect. 4.1) or *dynamically* from Web pages (*see* Sect. 4.2). After the ratings of the SDG are initialized, the system identifies the *first corpus of users* belonging to the SDG by matching their ratings with the *initialized* ratings of the SDG.

4.1 Initializing the Ratings of a SDG Statically

The preferences of a SDG can be acquired statically from *hard-copy* published studies such as:

(a) Published articles and books (e.g., [14, 25]).
(b) Published studies conducted by organizations (e.g., [9]), or specialized centers belonging to universities.

First, we need to decide on the publications to be used. The more publications used, the more accurate the results are. We need to select ones issued by reputable sources. Preferences on an item's features obtained from a hard-copy published study are represented as a publication-feature matrix M with entries f_i and P_j: feature f_i is recommended by publication P_j. The rating of publication P_j on feature f_i is the element $M(j, i)$ of matrix M. Element $M(j, i)$ is a Boolean value, where *one* denotes that publications P_j stresses the importance of feature f_i to the SDG and *zero* otherwise. That is, the rating $M(P_j)$ of publication P_j is the j-th row of matrix M. For example, consider the following *car* preferences of the residents of neighborhood N_x (i.e., *SDG*

Table 3 Publication-feature matrix M

	f_1	f_2	f_3	f_4
P_1	0	0	0	1
P_2	1	0	1	1
P_3	1	1	1	0
P_4	0	0	0	1
P_5	1	1	1	1
Score	3	2	3	4
Normalized score	0.25	0.17	0.25	0.33

N_x). N_x is a neighborhood in the State of Minnesota, USA. According to published surveys, 68 % of Minnesotans prefer cars with *snow-proof* features,[3] 61 % prefer *fuel-efficient* cars,[4] and 76 % of the residents of N_x prefer *cost-efficient* cars.[5] The preferences of N_x on each of these three features will be assigned a weight of *one* in matrix M. The score of a feature is the *summation of publications' weights on it* (*see* Eq. 1). Table 3 shows an example data set of matrix M. For example, the score of feature f_1 is the sum of the weights of publication P_2, P_3, and P_5 on feature f_1.

$$Score f_i = \sum_{j=1}^{|I|} M(P_j, f_i) \tag{1}$$

We now introduce an item-feature matrix N, where element $N(j, i)$ is *one*, if item I_j contains feature f_j and *zero* otherwise. The profile $N(I_j)$ of item I_j is the j-th column of matrix N. The score of item I_j is the *summation of the normalized scores* of the features that I_j contains (*see* Eq. 2)

$$ScoreI_j = \sum_{\forall N(f_i, I_j)=1} scoref_i \tag{2}$$

Table 4 shows an example data set of Matrix N. For example, the score of item I_1 is the sum of the normalized scores of features f_2 and f_3 which is 0.17 + 0.25 (*recall* Table 3). For maintaining the integrity of relativity, all *rival* features should be rated using the *same* publications.

[3] Due to the very snowy winter in the state of *Minnesota*.

[4] Which is due, in part, to the fact that the government of *Minnesota* offers sales tax break incentive for buying fuel-efficient cars.

[5] Due to the fact that N_x is a middle-class neighborhood (e.g., [16]).

Table 4 Feature-item matrix *N*

	I_1	I_2	I_3
f_1	0	1	0
f_2	1	1	0
f_3	1	0	1
f_4	0	1	0
Score	0.42	0.75	0.25

4.2 Initializing the Ratings of a SDG from Web Pages Dynamically by Crawling Web Sites

By crawling Web sites, XEngine initializes the preferences and ratings of SDGs dynamically from Web pages that publish information about them. We proposed previously in [22] an approach that identifies the semantic relationships between XML elements in an XML document. In Sect. 4.2.1, we describe modifications we made to [22] to suit the extraction of Web content data for the sake of dynamically initializing SDGs' preferences. In Sect. 4.2.2, we describe modifications we made to an approach proposed in [24] in order to initialize a SDGs' preferences. *The system generates items' scores by converting the preference data* (*obtained from the two approaches*) *into weighted web-feature and feature-item matrices* (*similar to* Tables 3, 4).

4.2.1 Employing the XCDSearch Approach in [22]

We proposed in [22] techniques called XCDSearch to build *semantic relationships* between elements in XML documents. For the sake of this work, we modified these techniques in order to build semantic relationships between Web content data (i.e., *instead of XML data*) to initialize the ratings of SDGs. We constructed a copy of XEngine that employs these techniques to dynamically identify the preferences of SDGs from Web pages that publish information about them. The system will then generate items' scores *dynamically* by converting this preference data to weighted *web-feature and feature-item matrices* (*recall* Tables 3, 4) using Eqs. 1 and 2. The system will use these matrices to *initialize* SDGs' ratings. First, the system will mark up a Web page with XML tags and model the resulting document as a rooted and labeled XML tree (e.g., Fig. 1). A SDG is represented as an interior node in the XML tree, and its preferences as data/leaf nodes. For example, Fig. 1 is a fragment of an XML tree modeling the content data of Web page publishing information about some SDGs.

We first define key concepts used in the modified techniques. We use the term Ontology Label to refer to the ontological concept of a node in an XML tree. Let (m "is-a" m') denote that class m is a subclass of class m' in an Object-Oriented ontology. m' is the most general superclass (root node) of m in a defined ontology hierarchy. m' is called the Ontology Label of m. The system converts an XML tree

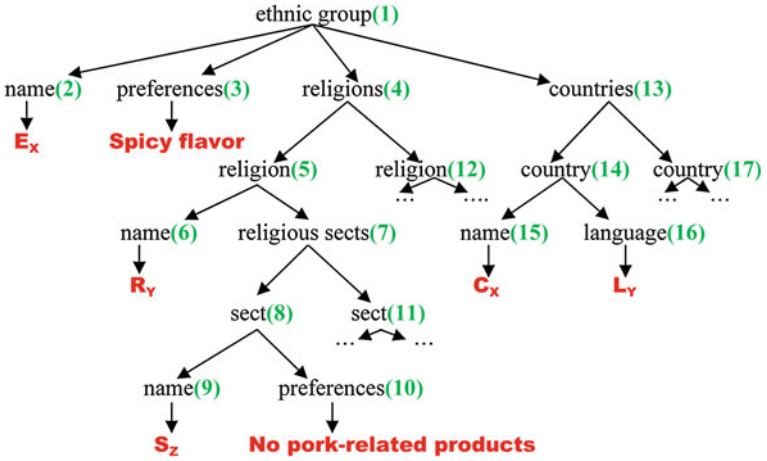


Fig. 1 A fragment of an XML tree modeling the content data of a Web page about some SDGs. *Nodes are numbered for easy reference*

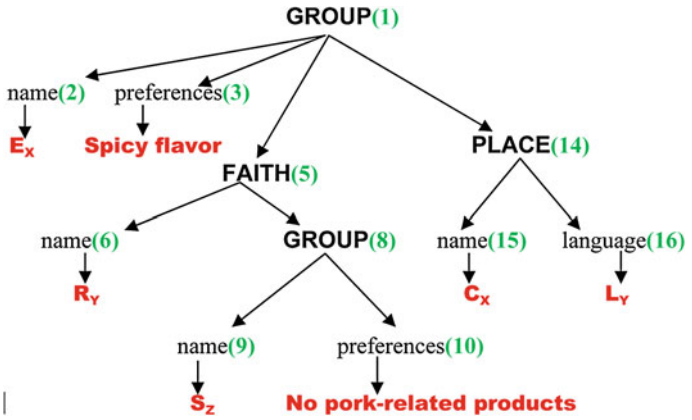


Fig. 2 Ontology-based tree constructed from the XML tree in Fig. 1

into a tree called ontology-based tree. For example, Fig. 2 shows an ontology-based tree constructed from the XML tree in Fig. 1. An ontology-based tree is constructed as follows. First, the system removes all interior nodes that do not have children data nodes (*for example, nodes 4, 7, and 13 are removed from* Fig. 1). Then, the system replaces the remaining *interior* nodes with their Ontology Labels (*for example, nodes ethnic group(1) and sect(8) in* Fig. 1 *are replaced by their Ontology Label, which is GROUP as shown in* Fig. 2).

Let a be an interior node and b a data node in an ontology-based tree. Nodes a and b are *semantically related* if the paths from a and b to their *Lowest Common Ancestor* (*LCA*), not including a and b, do not contain more than one node with

the same Ontology Label. The LCA of a and b is the only node that contains the same Ontology Label in the two paths to a and b. Consider that node b contains the preference data[6] P_i and that node a represents SDG G_j. If nodes a and b are semantically related, P_i is a preference of SDG G_j. For example, consider Fig. 2. Preference "*no pork-related products*" (node 10) belongs to religious sect R_Y (node 6) and not to ethnic group E_X (node 2), because the LCA of nodes 10 and 2 is node 1, and the path from node 1 to node 10 includes two nodes with the same Ontology Labels (i.e., *nodes 1 and 8*). Similarly, the preference "*spicy flavor*" (node 3) belongs to E_X and *not to* S_Z (node 9). Using the same techniques, both of "*spicy flavor*" and "*no pork-related products*" are preferences to religion group R_Y (node 6).

4.2.2 Employing the TopDown Approach in [24]

Tang et al. [24] studies the effect of *topic taxonomy* on dynamic *group profiling*. A topic taxonomy consists of topic nodes. Each internal node is defined by its vertical path (i.e., *ancestor and child nodes*) and its horizontal list of attributes. To perform taxonomy adaptation, the paper proposes a top-down hierarchical traversal approach called TopDown. We constructed a copy of XEngine that employs an *adjusted* version of the TopDown approach to identify and initialize the preferences of a SDG from Web pages publishing information about it. For each topic node n representing a SDG G_x, this copy of XEngine identifies the *best neighbor* nodes of n that contain preference data about G_x. The TopDown approach consists of multiple iterations to search for *better hierarchies*, as follows:

1. *Identification of the node to check*: A list of topic nodes in the hierarchy is maintained for the search. Nodes at the upper level are given higher priority.
2. *Identification of promising neighboring hierarchies concerning a node*: The promising hierarchies are checked by *rolling-up nodes* to their upper level. Then, the hierarchies are checked by *pushing down* nodes to their siblings and by *merging* two sibling nodes to form a super node.
3. *Identification of the best neighbor*: This procedure compares all the promising neighboring hierarchies and finds the best among them.
4. *Update of the current best hierarchy*: The current best hierarchy is replaced with the best hierarchy just found and the list of nodes to check is updated.

Example 2 Consider that the system crawled a web site publishing information about the Buddhism faith and identified the classificatory taxonomy of branches shown in Fig. 3. In the figure, p_x, p_y, p_z, and p_w are preference data. By *merging* nodes Mandalas and Shingon and *rolling up* the resulting node, and by *pushing down* node Mahayanists, the preferences of the Mahayanists can be identified as p_x, p_y, and p_z. By *pushing down* node Buddhists, it preferences can be identified as p_x, p_y, p_z, and p_w.

[6] The system identifies such data via *text mining program*.

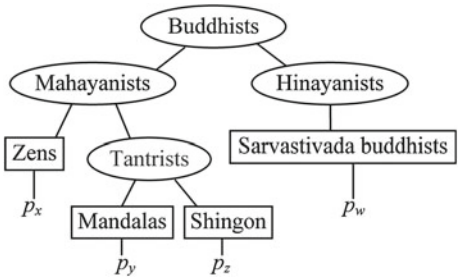


Fig. 3 Classificatory taxonomy of branches of the Buddhism faith

5 Identifying a SDG Members Implicitly and Updating a SDG Ratings Dynamically

5.1 Identifying Members of a SDG Implicitly

The system identifies (*implicitly*) new member users of a SDG G_x by matching their ratings with the rating pattern of G_x. Let D be the set of all domains. From *each* domain $d \in D$, the system will determine a SDG $\in d$ to which the user belongs, using a similarity measure. Let $sim(u_m, G_x)$ be the *similarity* between the preference vectors of user u_m and SDG G_x. The user u_m will be considered belonging to a SDG $G_x \in$ domain d_k if for each other SDG $G_y \in d_k$, $sim(u_m, G_x) > sim(u_m, G_y)$. We measure $sim(u_m, G_x)$ using the *cosine-similarity measure* shown in Eq. 3:

$$sim(u_m, G_x) = \frac{\sum_{\forall i \in I}\left(\left(r_{u_m,i} - \bar{r}_{u_m}\right)\left(r_{G_x,i} - \bar{r}_{G_x}\right)\right)}{\sqrt{\sum_{\forall i \in I}\left(r_{u_m,i} - \bar{r}_{u_m}\right)^2}\sqrt{\sum_{\forall i \in I}\left(r_{G_x,i} - \bar{r}_{G_x}\right)^2}} \tag{3}$$

- I: Set of features rated by SDG G_x and *co-rated* by u_m.
- $r_{u_m,i}$: Weight of user u_m on feature i.
- $r_{G_x,i}$: Normalized score of SDG G_x on feature i
- $\bar{r}_{u_m}$: Normalized *mean* weight of u_m on set I; $\bar{r}_{u_m} = \frac{\sum_{\forall i \in I} r_{u_m,i}}{|I|}$
- $\bar{r}_{G_x}$: Normalized *mean score* of G_x on set I; $\bar{r}_{G_x} = \frac{\sum_{\forall i \in I} r_{G_x,i}}{|I|}$

Equation 3 considers *each* feature rated by SDG G_x and co-rated by user u_m even if the feature was rated by only one member of G_x. Therefore, the equation *may give misleading similarity results*, since some features in set I may not reflect the actual preferences of G_x. A feature that has been rated very low or by few members of SDG G_x is most likely rated by a member(s) of G_x who belongs also to another SDG G_y. That is, this member(s) may belong to a MDG composed of G_x and G_y. This feature is likely reflects a preference of G_y and not G_x. That is, a user who belongs to a MDG $\{G_x, G_y\}$ is most likely to rate features that reflect the preferences of both G_x and G_y. Therefore, when measuring the similarity between an active user and G_x, we should consider only the features that reflect the preferences of G_x. That is, we

need to consider only the dominant features of G_x (i.e., *the features that have been rated high and by the majority of the members of* G_x). Consider for example that 10 % of the members of G_x belong to also another SDG G_y. The scores of G_x on features reflecting the preferences of G_y are most likely be low, since 90 % of the members of G_x may not rate them high. Intuitively, these are *non-dominant* features for G_x.

We adopt the following strategy for determining the set of dominant features for a SDG. From the set F of all features, the subset F' is the dominant features for a SDG, if every feature in F': (1) dominates every feature not in F_i (i.e., *has a greater score*), and (2) acquires a score greater than a threshold z. For example, recall Table 2 and consider that z is set to "+1". Accordingly, the set F_i of dominant features for the SDG would be $\{f_2, f_9, f_5, f_{10}\}$. We now formalize the concept of dominant features.

Definition 5.1 ***Dominant features for a SDG***: Let F be a set of n features and c(f) be the score of feature f. The subset $F' \subset F$ of dominant features with maximal scores for a SDG is given by: $\{a \in F{:}c(a) \geq c(b)$, for all $b \in F\}$ and $\{c(a) > z{:}(n-1) > z < -(n-1)\}$.

We adjusted Eq. 3 so that only the subset $F' \cap I$ is considered, as shown in Eq. 4.

$$sim(u_m, G_x) = \frac{\sum_{\forall i \in F''} \left(\left(r_{u_m,i} - \bar{r}_{u_m}\right)\left(r_{G_x,i} - \bar{r}_{G_x}\right)\right)}{\sqrt{\sum_{\forall i \in F''} \left(r_{u_m,i} - \bar{r}_{u_m}\right)^2} \sqrt{\sum_{\forall i \in F''} \left(r_{G_x,i} - \bar{r}_{G_x}\right)^2}} \tag{4}$$

- F': Set of dominant features rated with maximal scores by SDG G_x
- F'': Subset of F' co-rated by u_m (i.e., $F'' \subseteq F'$).
- $\bar{r}_{u_m} = \frac{\sum_{\forall i \in F''} r_{u_m,i}}{|F''|}$ and $\bar{r}_{G_x} = \frac{\sum_{\forall i \in F''} r_{G_x,i}}{|F''|}$

From the set F'', Eq. 4 overlooks the subset $F' - F''$ (i.e., *the subset that has not been co-rated by user* u_m). Therefore, the equation may give inaccurate similarity results. We observe that we can consider user u_m assigned a weight of zero to each of the features in the subset. The reason is that users usually have either no or very little interest on features they do not rate. We adjusted Eq. 4 to consider the subset $F' - F''$ *as shown in* Eq. 5.

$$sim(u_m, G_x) = \frac{\sum_{\forall i \in F''} \left(r_{u_m,i} - \bar{r}_{u_m}\right)\left(r_{G_x,i} - \bar{r}_{G_x}\right) + \sum_{\forall j \in P} \left(\bar{r}_{u_m}\left(r_{G_x,j} - \bar{r}_{G_x}\right)\right)}{\sqrt{\sum_{\forall i \in F''} \left(r_{u_m,i} - \bar{r}_{u_m}\right)^2 + |P|\left(\bar{r}_{u_m}\right)^2} \sqrt{\sum_{\forall i \in (F'' \cup P)} \left(r_{G_x,i} - \bar{r}_{G_x}\right)^2}}$$
$$P = \{F' - F''\} \tag{5}$$

Let F_u be the set of features rated by user u_m. As a final improvement of the similarity equation, we consider each feature $f_k \in \{F_u - F'\}$, if the weight of SDG G_x on f_k beat other features' weights at least k number of times, where $k > 0$. However, we need to penalize each expression operand in the equation involving f_k to ensure that it will have a lower impact on the similarity result. Moreover, we need to scale down these expressions appropriately to account for the rank specificity of f_u among the list of features ranked by G_x to ensure that that lower ranked features

indeed get higher penalty. Towards this, we *penalize and scale down* each expression operand involving f_u by a factor $decay^{t-1}$, where *decay* is a parameter that can be set to a value in the range 0 to 1. We set the exponent t to account for the rank of f_u among the list of features ranked by SDG G_x. *We adjusted* Eq. 5 *accordingly as shown in* Eq. 6.

$$sim(u_m, G_x) = \frac{\sum_{\forall i \in F''} \left(r_{u_m,i} - \bar{r}_{u_m}\right)\left(r_{G_x,i} - \bar{r}_{G_x}\right) + \sum_{\forall j \in P} \left(\bar{r}_{u_m}\left(r_{G_x,j} - \bar{r}_{G_x}\right)\right) + N}{\sqrt{\sum_{\forall i \in \{F'' \cup F_k\}} \left(r_{u_m,i} - \bar{r}_{u_m}\right)^2 + |P|\left(\bar{r}_{u_m}\right)^2}\sqrt{\sum_{\forall i \in (F'' \cup P)} \left(r_{G_x,i} - \bar{r}_{G_x}\right)^2 + M}} \tag{6}$$

$$N = \sum_{\forall i \in V} \left(r_{u_m,i} - \bar{r}_{u_m}\right)\left(r_{G_x,i} - \bar{r}_{G_x}\right) \times decay^{t-1}$$

$$M = \sum_{\forall i \in V} \left(r_{G_x,i} - \bar{r}_{G_x}\right)^2 \times decay^{t-1}$$

$$V = \{F_k - F'\}$$

F_k: Set of features that are: (1) rated by G_x, (2) co-rated by u_m, and (3) assigned weights by G_x that beat other features' weights at least k number of times.

Example 3 Recall Table 2. Consider that the threshold k in Eq. 6 has been set to 4. Thus, feature f_3 will be considered in Eq. 6, if the active user co-rated it (*even though* $f_3 \notin F'$. The expressions in Eq. 6 involving f_3 will be penalized by a factor decay^{t-1}, where t is 5 (i.e., *the rank of* f_3 *in the set rated by SDG* G_x). Parameter *decay* can be set to a value from 0 to 1.

5.2 Optimizing and Updating the Ratings of a SDG

As *each* new user is identified by the system as belonging to a SDG G_x (*using* Eq. 6), the current ratings of G_x will be re-optimized and re-updated (*dynamically*) based on the ratings of: (1) this new user, and (2) other member users of G_x. The ratings of the first corpus of users would update and optimize the *initialized* features' scores for the *SDG* (*recall* Sect. 4). The ratings of *each* subsequent user would *update* and *optimize* current features' scores for the SDG by updating: (1) features' number of beats/looses and scores (*recall* Table 2), and (2) the set of dominant features for the SDG (*recall* Definition 5.1). The preferences of a SDG G_x are stored in the system in the form of a *trigger rule*, called *TrigRule*(G_x). The computation time complexity of updating and optimizing current ratings of a social group is not expensive, because only the beats/looses matrix needs to be updated. Moreover, the computation complexity can be improved by updating the matrix after a certain number of new member users join the social group rather than updating it after each new member user joins the social group.

Definition 5.2 ***TrigRule***(G_x): *TrigRule*(G_x) is a *construct of rules* formed from predicate Boolean conditions. These conditions represent the preferences of SDG G_x. In response to a query submitted by a user belonging to G_x, *TrigRule*(G_x) filters

XML tuples, *retaining only* those satisfying the preferences of G_x. The construct of a trigger rule is formed from the "*WHERE*" clause of XQuery [6].

A trigger rule has the form [$\$b \Delta L \Diamond P$ *and*/*or*/*not*…$\$b \Delta L \Diamond P$]. The symbol Δ denotes either an XQuery's child '/' or descendant operator '//'. The symbol $\Diamond$ denotes an XQuery comparison operator. P denotes a SDG's *preference data* contained in the XML document in a data element labeled L.

Example 4 Consider that after applying the strategy described in Definition 5.1, the following food preferences are identified: (1) feature "spicy flavor" is a preference for ethnic group E_y, and (2) feature "no pork-related products in ingredients" is a preference for religious group R_x(e.g., *because the teachings of religion* R_x *dictate that*). These preferences will be stored as the following *trigger rules*: *FoodTrigRule*(E_y) {*$b*/flavor = "spicy"}; *FoodTrigRule*(R_x) {contains (*$b*/ingredients, "no pork-related products")}.

6 Modeling SDGs and Identifying the User's Smallest MDG Implicitly

6.1 Modeling SDGs

XEngine identifies the user's smallest MDG (*recall* Definition 3.3) by modeling the *relationships* between SDGs using an *ontology-driven* graphical representation called *Single Domain Graph* (*SDGraph*). A SDGraph represents the *domain ontology* of SDGs. Let U_s be the set of users belonging to *both* SDGs G_x and G_y, whose *domains are different* (i.e., $U_s \subseteq G_x, G_y$). If $|U_s| > m$, (*where m is a threshold parameter determined by the System Administrator*), G_x and G_y are related and their vertices in the SDGraph will be linked by an edge to represent this relationship. That is, if the number of users who are members of *both* SDGs is greater than m, the vertices of the two SDGs will be linked by an edge in the SDGraph. If G_x and G_y are not linked by an edge, but each is linked by an edge with another SDG G_z, one can infer that G_x and G_y are related by being superclasses of G_z.

Definition 6.1 ***Single-Domain Graph*** (***SDGraph***): A SDGraph is a pair of sets (V, E), where V is a finite set of vertices representing SDGs and E, the set of edges, is a binary relation on V, so that $E \subseteq V \times V$. $E = \{edge(u, v)$:$u \in V$ *and* $v \in \pi[u]\}$, where $\pi[u]$ is the set of vertices representing the immediate subclasses of the SDG represented by the vertex $u \in V$.

Example 5 Consider the following USA-based SDGs of four domains: (1) *ethnic* groups E_X and E_Y, (2) *religious* groups R_X and R_Y, (3) *national origin* group O_X, and (4) *region-based* groups N_X and N_Y (*the people living in neighborhoods* N_X *and* N_Y *respectively*), Ethnic group E_Y lives in neighborhood N_X, and follows religion

R_Y. Ethnic group E_X lives in neighborhoods N_X and N_Y. Part of E_X follows religion R_X, and the other follows R_Y. Figure 4 shows a SDGraph constructed based on the above information.

Internally, XEngine constructs a SDGraph as follows. It first prompts the system administrator with two GUI text fields, one represents a SDG G_x and the other represents the *immediate-subclass* of G_x. The system administrator can acquire this data from published studies, such as: (1) government censuses and statistical studies (e.g., [26]), and (2) published studies conducted by specialized centers in universities and organizations (e.g., [8]). *After receiving the SDGs' data, XEngine will represent the data internally as OWL ontology* [4]. *The OWL file defines the relations between SDGs as ontological classes.*

6.2 Identifying the User's Smallest MDG Implicitly

XEngine provides the user with a GUI to reveal his *initial* preferences (*represented by his/her weights on features*). Based on these ratings, the system identifies the user's SDGs and smallest MDG. A user may belong to more than one MDG. The smaller a MDG is, the more granular its interests are. Therefore, the system determines the user's smallest MDG to enable it to return the most relevant results. To determine the user's smallest MDG, the system needs first to identify *at least one of the SDGs* to which the user belongs. XEngine adopts the following approaches for identifying the user's SDGs: (1) it matches his rating with the ratings of SDGs using Eq. 6 (*recall* Sect. 5.1), (2) it employs lookup databases provided by Maponics [15] to *implicitly* identify all *region-based* SDGs (*by translating users' IP addresses to US neighborhoods, cities, and states*), and (3) it analyzes the structure of the SDGraph.

> For example, by analyzing the structure of Fig. 4, if the user lives in neighborhood N_X and follows religion R_Y, the system *can determine implicitly* that he/she belongs to ethnic group E_Y and national origin O_X.

After identifying some of the SDGs to which the user belongs, XEngine identifies the smallest MDG to which the user belongs by *traversing the paths of the SDGraph*, as follows. The user's smallest MDG is formed from: (1) SDGs located in the paths from the user's known SDGs to their lowest common descendant, and (2) SDGs located in the path from the lowest common descendant to the lowest SDG. If the paths originated from n number of vertices (i.e., *n number of SDGs*), MDG is usually formed from m number of SDGs, where $m > n$. Therefore, the *more* SDGs identified for the user, the *smaller in size* a MDG can be identified (*recall that the more SDGs a MDG is composed of, the smaller in size it becomes*).

Example 6 Consider Fig. 4 and that the system *identified through the lookup databases* that the user's neighborhood is N_X. Consider also that the system identified *implicitly* the user's religion as R_Y (*by matching the user's ratings with the ratings of* R_Y *using* Eq. 6). Accordingly, the system can determine that the user's smallest MDG

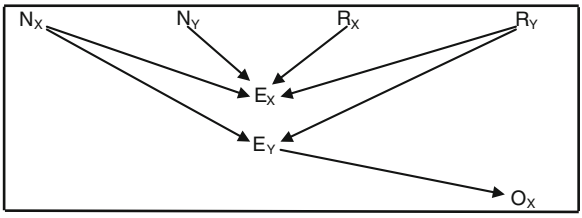


Fig. 4 A SDGraph depicting relationships between some USA SDGs

```
FilterResults (Q, V_Y) {
1.   SItems ← GetInitialResults(Q)
2.   for each SDG  G_x ∈ V_Y
3.        SItems ← RefineSelection(G_x , SItems)
}
```

Fig. 5 Algorithm *FilterResults*. Subroutine *RefineSelection* iterates over the results in set *SItems*. It filters the results recursively. Each recursion filters the results based on the preferences of a SDG $\in$ MDG V_Y

is $\{N_X, R_Y, E_Y, O_X\}$. This is because: (1) E_Y is the lowest common descendant of N_X and R_Y, and (2) O_X is a descendant of E_Y. As can be seen, the system started the search using two SDGs and it could locate a MDG composed of four SDGs.

7 Filtering and Ranking Results

In response to a user's query/ratings, the system first constructs an XQuery query [6] *equivalent* to the user's query to get the *initial results*. After identifying the user's smallest MDG, XEngine uses an algorithm to filter and rank the user's *initial results* based on the preferences/ratings of the SDGs composing the user's smallest MDG. The algorithm is called *FilterResults* (see Fig. 5). It is an *XML-based* framework. It employs recursive querying to sequentially optimize (*filter*) results. In each optimization sequence, the results are filtered based on the preferences of one of the SDGs forming the user's smallest MDG. That is, in *each optimization sequence* the algorithm triggers trigger rule TrigRule(G_x) where G_x is one of the SDGs forming the user's smallest MDG.

The inputs to Algorithm *FilterResults* are: (1) an XQuery query Q *equivalent* to the user's query, and (2) the user's smallest MDG V_Y. In line 1 of the Algorithm, function *GetInitialResults* uses an XQuery *search engine* to return the *IDs* of the items that satisfy the conditions of query Q. These *IDs* will be stored in a set called *SItems*. Line 3 calls an *XML-based* subroutine called *RefineSelection* to *recursively* filter the *IDs* in set *SItems*. That is, the subroutine iterates over the *IDs* in set *SItems*. In each iteration, the set is passed to the subroutine as a parameter; the return of the

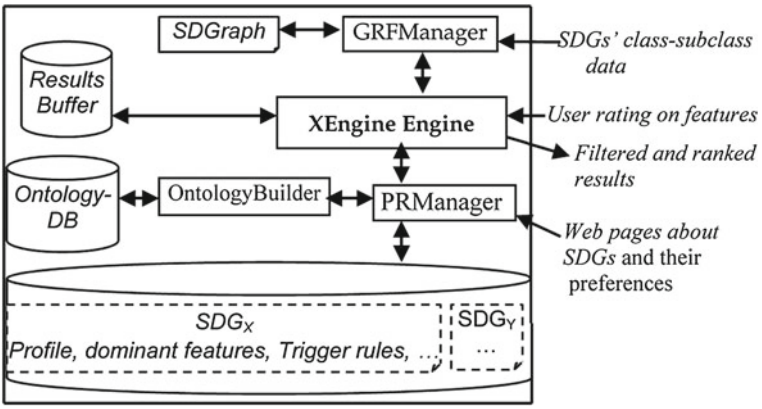


Fig. 6 System architecture

subroutine will be stored *in the same set*. In each iteration, *only* the IDs belonging to items satisfying the preferences of a SDG $G_x \in V_Y$ are retained. The subroutine includes FLWOR *expressions* of XQuery [6]. The "where" clause of FLWOR filters items in the XML document retaining only those whose: (1) *IDs* match the *ID* from set *SItems*, and (2) elements contain preference data matching the preference data triggered by trigger rule TrigRule(G_x), where G_x is *one of the SDGs* composing the user's MDG.

After the items have been *filtered*, they will be *ranked* based on the *vector D of scores* rated by the SDGs composing the user's smallest MDG. Items whose scores are high are *ranked higher*. A score T_x of an item is defined as the summation of the *normalized scores* on the item's features rated by all the SDGs composing the user's smallest MDG.

$$T_x = \sum_{j=1}^{n} d_{i,j} \tag{7}$$

- $d_{i,j}$: Score on feature i rated by SDG j; $d_{i,j} \in D$
- n: *Number of SDGs* composing the *user's smallest MDG*

8 System Architecture

Figure 6 shows the *system architecture*. Module GRFManager performs the following: (1) converts SDGs' *class-subclass* data into a SDGraph, and (2) identifies all possible MDGs that exist because of the interrelations between SDGs.

Module PRManager performs the following for *each* SDG G_x: (1) *initializes* the ratings of G_x either dynamically from Web pages or statically from published studies, (2) for the dynamic initialization, it uses *Protégé ontology editor* [19] to build

Ontology Labels and saves the ontologies in database Ontology-DB, (3) *optimizes* and *updates* the ratings of G_x based on the ratings of new *member users*, (4) stores the preferences of G_x in the system's database in the form of a trigger rule, (5) assigns a folder for G_x in the system's database to store information such as the name, ratings, and preferences of G_x, and (6) assigns a subfolder *within* the folder of G_x for each user belonging to G_x to store information such as the ratings of the user. That is, individual subfolders are associated with G_x folder via file in a *file system structure in a two-layer directory tree*.

XEngine engine performs the following: (1) constructs an XQuery query [6] *equivalent* to the user's query/ratings and gets the *initial results* using an *XQuery search engine*, (2) identifies *implicitly* the user's SDGs and smallest MDG by consulting module PRManager, and (3) *filters* and *ranks* results based on the ratings of the smallest MDG, to which the user belongs.

9 Experimental Results

We implemented XEngine in Java and ran it on an Intel(R) Core(TM)2 Dup CPU processor, with a CPU of 2.1 GHz and 3 GB of RAM, under Windows Vista. Our objective is to compare: XEngine, FWNB [21], Amazon Elastic MapReduce [2], and SPGProfile [23], using: (1) MovieLens data set [17], (2) simulated-user evaluation,[7] and (3) real-user evaluation.[8]

9.1 Methodologies for Obtaining Test Data Sets

9.1.1 MovieLens Data Set

MovieLens is a movie recommender system maintained by GroupLens Research [17]. MovieLens data set contains 10 million ratings on 10,000 movies provided by 72,000 registered users. We extracted the content features (e.g., *actors, directors, keywords*) and the ratings on these features from the internet movie database (imdb) [13]. After joining the imdb and Movielens data sets, we identified 2,664 actors and actresses, 12,076 keywords, and 1,163 directors from 1,700 movies. To conform to the experimental protocol followed in [21], we selected only the *3 best paid actors/actresses* for each movie.

For the sake of the experiments, we assume the following analogies concerning XEngine. Each group of users who like a specific film genre forms a SDG

[7] We simulate an online grocery dealer running the four systems and targeting *20 USA neighborhoods*; this evaluation will be conducted using *test data* acquired from *real individuals* living in the 20 neighborhoods.

[8] This evaluation is conducted by students from UTA.

(e.g., $SDG_{classis}$). A user who likes two or more genres, is considered to belong to a MDG composed of the genre-based SDGs, which the user likes. For example, a user who likes both classic and horror movies is considered to belong to MDG = {$SDG_{classis}$, SDG_{horror}}. We *initialized* the preferences and ratings of the genre-based SDGs using the copy of XEngine that employs the modified version of XCD-Search (*recall* Sect. 4.2.1). Towards this, the system crawled 30 Web sites publishing information about top-rated movies per genre. It identified these movies using the approach and text mining technique described in Sect. 4.2.1. A movie is considered top-rated if it has been rated at least 6/10. For each genre-based SDG, the system generates its scores on movies by creating weighted web-feature and feature-item matrices (*similar to* Tables 3 and 4).

We then submitted the ratings of the registered member users of [13, 17] to FWNB [21] and XEngine. FWNB grouped the users into biclusters (*recall* Sect. 2). As for XEngine, it populated the 22 genre-based SDGs[9] with the users. The ratings of the first corpus of users from [13, 17] updated and optimized the *initialized* ratings of the genre-based SDGs. The ratings of each subsequent user from [13, 17] would update and optimize *current* ratings/scores of a SDG.

9.1.2 Test Data Set for Simulating User Evaluation

We aimed at obtaining *ethnic* and *religious-based* food preference data from *real* individuals living in 20 USA neighborhoods in order to further evaluate XEngine. Towards this, we selected 20 USA neighborhoods known for their ethnic and religious diversity. Each neighborhood represents a *region-based* SDG. From the combined residents of the 20 neighborhoods, those who belong to a same ethnicity represent an *ethnic-based* SDG, and those who have the same religious beliefs represent a *religious-based* SDG. After obtaining the food preference/rating data from people belonging to the above described SDGs, we would load the data/ratings into the four systems. Then, we would determine which system(s) returns ranked list of canned food that *matches closely* to the ones ranked by the subjects living in the 20 neighborhoods.

We needed to identify some of the residents living in the 20 USA neighborhoods to acquire their food preferences. Using an online phone book, we selected some of these residents. Along with six volunteers, we phoned them, explained the objectives of our research, and asked them to: (1) provide us with their *ethnicities* and *religions*, and (2) *rate* their preferences (and restrictions) on five different canned soup flavors and ingredients in the scale of 1–10. Some of the residents declined to provide us with the information, while others agreed to. From over 600 calls, only 278 residents provided us with the information. The 278 residents included *at least 10* ones from *each* of the 20 neighborhoods. Table 5 shows the number of *region*, *ethnic*, and *religious* SDGs constructed from the acquired data.

[9] By matching the ratings of each member user from [13, 17] with the ratings of the genre-based SDGs, using Eq. 6 (recall Sect. 5).

Table 5 Number of region, ethnic, and religion SDGs from which we acquired preference data

State	MN	TX	WA	FL	CA
City	MPLS	Dallas-Fort Worth	Seattle	Miami	Los Angles
Number of neighborhood SDGs	5	4	2	3	6
Number of ethnic-based SDGs	3	4	3	4	4
Number of religious-based SDGs	2	4	2	3	4

9.1.3 Test Data for Real-User Evaluation

We asked 32 students from The University of Texas at Arlington (UTA) to evaluate and compare the four systems. The students belong to four different ethnic backgrounds and five ancestry origins. Some of them consider religion to be irrelevant and the others follow three different religions. We asked each of the students to prepare a list of 10 canned food items *ranked* based on the student's *own preferences*. We then asked this student to query each of the four systems for canned food to determine which one(s) returns ranked list of canned food *matches closely* to the one ranked by the student himself/herself.

9.2 XEngine Demo System

A demo of the XEngine system targeting the 20 USA neighborhoods (*described in* Sect. 9.1.2) is available at: http://dbse1.uta.edu/kamal/XEngine/. The demo filters results based on the *preferences/ratings* of the 278 residents of the neighborhoods described in Sect. 9.1.2. For more information about the demo system, click on the tab "ABOUT XEngine Demo" in the demo's Web site. The XEngine system and the other three systems we are going to compare run on 1,000 MBs *grocery.xml* document containing 28233719 XML nodes. *The data for the grocery.xml document is obtained from* [3].

9.3 Comparing Three Approaches for Initializing the Ratings of a SDG

We compare in this test the three approaches described in Sect. 4 for *initializing* the preferences and ratings of a SDG. These approaches are: (1) the *static initialization* from *hard-copy* published studies (*recall* Sect. 4.1), (2) the *dynamic initialization* using the modified version of XCDSearch [22] (*recall* Sect. 4.2.1), and (3) the *dynamic initialization* using the modified version of TopDown [24] (*recall* Sect. 4.2.2). We cloned the XEngine system into three identical copies, each employing one of the three approaches described above. Our objective is to determine which

one of the three copies gives ranked lists of canned food *closest* to those ranked by the 278 subjects living in the 20 US neighborhoods (*recall* Sect. 9.1.2). For the experimental dataset, we selected 18 Web sites (e.g., [10, 29]) publishing information about social groups and their preferences. For the sake of consistency, we used the same dataset for evaluating the static initialization approach also (*rather than using published hard copies*).

We ran the Web pages (*dynamically*) against each of the two copies employing the *dynamic approaches*. As for the copy employing the *static initialization* approach, we entered the preference data from the Web pages *manually* into the copy. We then measured the distance $d(\sigma_u, \sigma_s)$ between each list ranked by a resident u and the corresponding list ranked by one of the three copy systems s, using the following Euclidean distance measure.

$$d(\sigma_u, \sigma_s) = \sum_{x \in \chi} |\sigma_u(x) - \sigma_s(x)| \tag{8}$$

X: Set of canned food items.
$\sigma_u \in [0, 1]^{|\chi|}$: List of items ranked by resident u.
$\sigma_s \in [0, 1]^{|\chi|}$: A list ranked by *one of the three copy systems*
$\sigma_u(x)$ and $\sigma_s(x)$: *position* of canned food item $x \in X$ in the lists σ_u and σ_s respectively (*a ranking of a set of n items is represented as a permutation of the integers* $1, 2, \ldots, n$).

Intuitively, the *static initialization approach* is expected to be more accurate than the other two approaches, since data is entered to the system *manually*. However, we aim at studying: (1) how much less accurate are the dynamic approaches than the static approach and *whether this accuracy difference is significant*, (2) whether the *practicality and convenience* of the dynamic approach makes up for its lower accuracy, in case the accuracy difference is not significant, and (3) the impact of *number of publications* on the accuracy of the three approaches. Figure 7 shows the results. We can infer from the results the following:

1. The static approach outperforms the two dynamic approaches as long as *the number of publications is less than about 25*.
2. The XCDSearch's approach outperforms the TopDown approach *as long as the number of publications is greater than about 10*.

Based on the experiment results, we advocate employing the XCDSearch's approach for the sake of practicality and dynamicity, especially for recommender systems that target a rather wide range of SDGs.

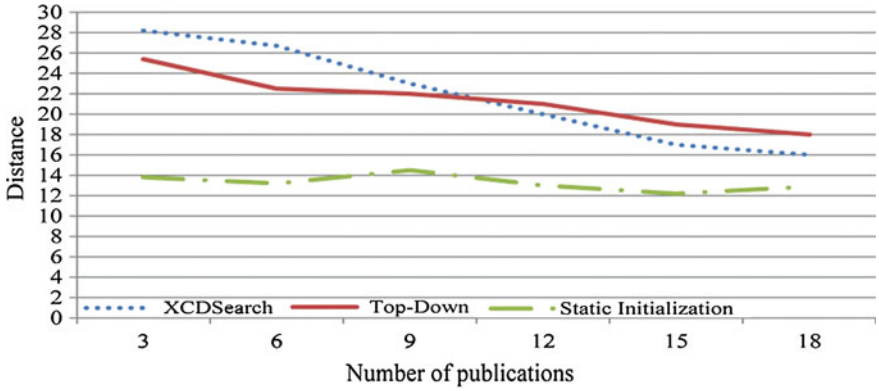


Fig. 7 Distance between the lists of items ranked by the USA residents and the lists ranked by the copies of XEngine employing the XCDSearch, TopDown, and static initialization approaches

9.4 Evaluating the Effectiveness of XEngine to Update and Optimize the Ratings of a SDG

We evaluate in this test the effectiveness of XEngine to *update* and *optimize* the preferences and ratings of a SDG (*after being initialized*) by comparing it with FWNP [21] (*recall the description of FWNP in* Sect. 2). Our objective is to determine whether XEngine or FWNP gives ranked lists of items closer to those ranked by: (1) MovieLens members (*recall* Sect. 9.1.1), (2) the 278 residents living in the 20 USA neighborhoods (*recall* Sect. 9.1.2), and (3) the 32 UTA students (*recall* Sect. 9.1.3).

9.4.1 Evaluation Using MovieLens Data Set

In this test, we evaluate XEngine by comparing it with FWNB using the same experimental protocol that [21] followed for evaluating FWNB. Towards this, we compare the two approaches using: (1) MovieLens Data Set (*recall* Sect. 9.1.1), which was used in [21], and (2) the same metrics used in [21], which are recall, precision, and explain coverage. Let: (1) N be the number of movies in a recommended list, (2) R_n be the number of relevant movies for the user in the recommended list that are rated higher than a threshold p by the user, and (3) R_{ALL} be the total number of relevant movies for the user. Recall = R_n / R_{ALL} and Precision = R_n / N. Explain coverage measures the number of movie features that are: (1) rated by the user to a value greater or equal to the threshold p, and (2) covered by the features in the movies recommended by the system. To conform to the experimental protocol followed in [21], we set the default size N to 20 and the rating threshold p to 3.

Figure 8 shows the recall-precision diagram. As the figure shows, XEngine outperformed FWNB. It outperformed FWNB in precision because: (1) FWNB forms a

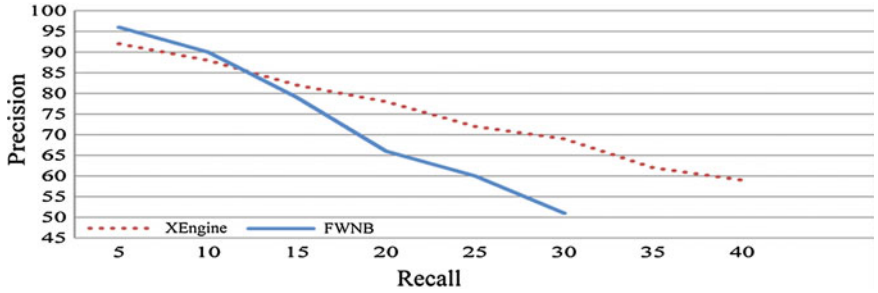


Fig. 8 Comparing XEngine with FWNB in recall versus precision using MovieLens

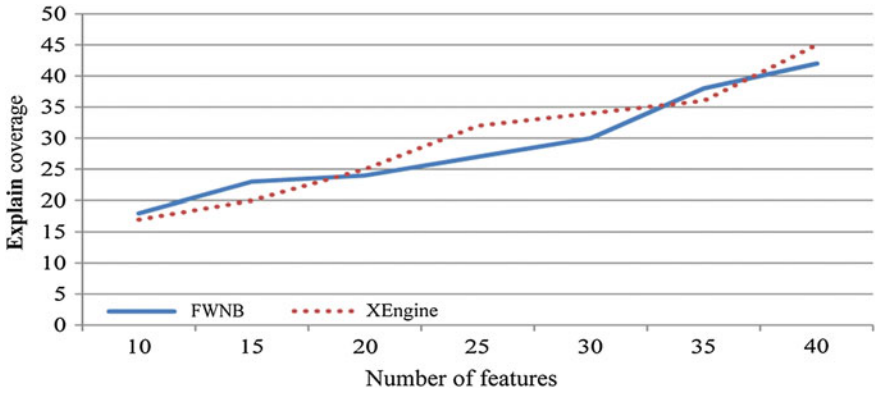


Fig. 9 Comparing XEngine with FWNB in explain coverage versus number of features using MovieLens

bicluster based on the rating similarity of its members on *items*, while XEngine forms a SDG based on the rating similarity of its members on the *features* of the items, (2) FWNP ranks items based on the frequency of the items' features in the profile of a bicluster, which may not be always accurate,[10] while XEngine ranks items based on the ratings of the user's smallest MDG, and (3) the effectiveness of XEngine's group modeling strategy and similarity equation. XEngine outperformed FWNB in recall because FWNB considers *only* items rated by the active user and co-rated by a bicluster, while XEngine considers: (1) *all* dominant features of the user's SDGs, even if the user did not co-rate some of them, and (2) *non-dominant* features of the user's SDG, whose assigned weights beat other features' weights at least k number of times.[11] Figure 9 shows explain coverage versus number of recommended list N. As the figure shows, intuitively, explain coverage increases as N increases. None of XEngine and FWNB shows clear performance over the other.

[10] For example if the active user's favorite actor acted in very few movies, FWNP will most likely not include this actor's movies in its recommendation list, or may rank them low.

[11] In the experiments, we set the threshold k to (*number of features*)/2.

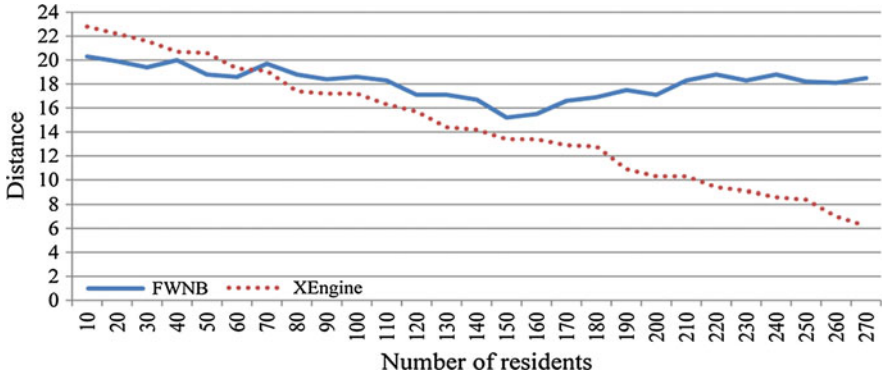


Fig. 10 The distances between the lists of items ranked by the USA residents and the lists ranked by FWNB and XEngine

9.4.2 Evaluation via the Simulated User Evaluation

We submitted the *ratings* provided by the 278 USA residents to both XEngine and FWNP [21] to *simulate* their evaluation of the systems. We calculated the average *cumulative* Euclidean distance between the lists ranked by the residents and the corresponding ones ranked by a system *within each interval of 10 residents*, using Eq. 4. Let n be a *cumulative* of 10 residents (n = 10, 20, …, 278). We measured the average *cumulative* Euclidean distance between the lists ranked by *each n* residents and the corresponding ones ranked by a system. Figure 10 shows the results. We can infer from the experimental results that:

1. The "closeness" between the lists ranked by the residents and the lists ranked by XEngine *increases consistently* as the cumulative number of residents increases. This is because after the ratings of *each* resident are submitted to XEngine, it *updates* and *optimizes* the current ratings of the resident's SDGs based on the ratings of this resident.
2. The distances of FWNB are *not* impacted by the cumulative increases of the *number of residents/ratings*.
3. As the residents' ratings were being submitted to the two systems, FWNB was outperforming XEngine until the cumulative number of these ratings reached about 70. Thereafter, XEngine kept *consistently* outperforming FWNB. Thus, for a *practical* application (*such as a business that markets preference-driven products*), XEngine can be preferable to FWNB.

9.4.3 Evaluation via Real-User Evaluation

We asked the 32 UTA students (*recall* Sect. 9.1.3) to calculate the average *cumulative* Euclidean distance between the lists they ranked and the lists ranked by the two sys-

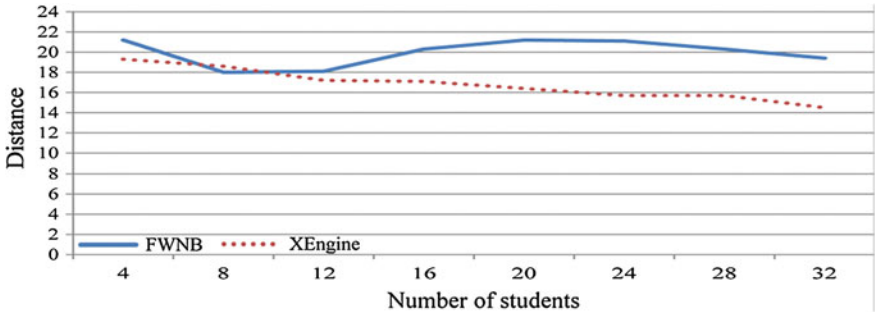


Fig. 11 Distance between the lists ranked by the students and the lists ranked by FWNB and XEngine

tems *within each interval of 4 students*, using Eq. 8. Figure 11 shows the results. The results concurred with the results of the simulated evaluation described in Sect. 9.4.2, where XEngine outperformed FWNB.

9.5 Comparing XEngine with SPGProfile

In this test, our objective is to compare the accuracy of the *dynamic-implicit techniques* and *static-explicit techniques* of group profiling. Towards this, we compared XEngine (*as a representative of the former approach*) with SPGProfile [23] (*as a representative of the later approach*). Intuitively, we expected the *static-explicit* approach to be more accurate, since: (1) users reveal *explicitly* to the system their SDGs, and (2) the preferences of SDGs are determined *statically*. *We aim at studying how significant is the accuracy difference between the two approaches*. If not significant, the *practicality and convenience* of the dynamic-implicit approach would *make up* for its lower accuracy and would *pay off*.

We submitted to SPGProfile the *initial preferences* and *SDGs' names* of the 278 USA residents and calculated the average *cumulative* Euclidean distance between the lists ranked by the residents and the ones ranked by SPGProfile, using Eq. 4. Figure 12 shows the results. We also asked the 32 UTA students to calculate the average *cumulative* Euclidean distance between the lists they ranked and the lists ranked by SPGProfile, using Eq. 4. Figure 13 shows the results. As Figs. 12 and 13 show, SPGProfile outperformed XEngine (*as expected*). However, this performance is *slight*, which is an indicative that there is *no significant* decline in the search quality of the *dynamic-implicit* approach compared to the *static-explicit* approach. *Therefore, we believe that the practicality and convenience of the dynamic-implicit approach can make up for the approach's slight lower accuracy*.

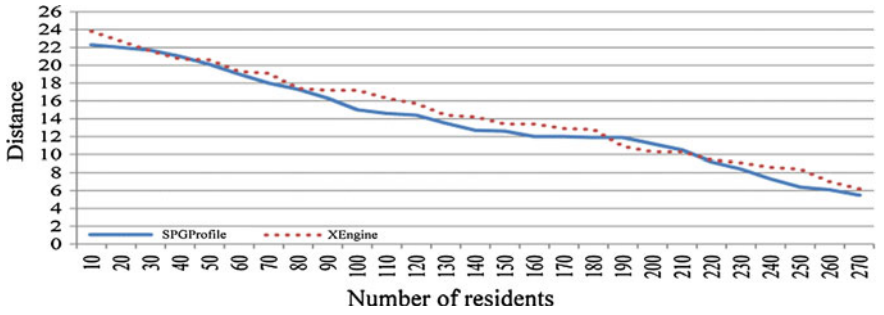


Fig. 12 Distance between the lists of items ranked by the residents and the lists ranked by SPGProfile and XEngine

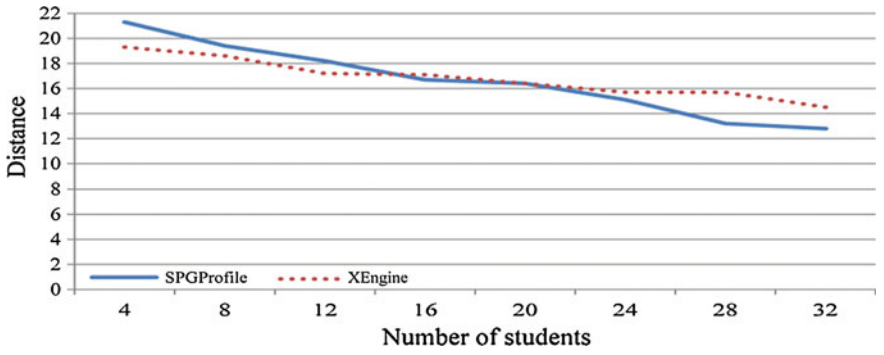


Fig. 13 Distance between the lists ranked by the students and the lists ranked by SPGProfile and XEngine

9.6 Comparing CF Employing Group-Based Profiling and User-Based Profiling

Our objective is to determine whether CF *employing group-based* profiling techniques or CF *employing user-based* profiling techniques gives ranked lists of items closer to those ranked by users themselves. Towards this, we compared *both* XEngine and FWNB (*as representatives of the former approach*) with Amazon Elastic MapReduce [2] (*as a representative of the later approach*). Amazon Elastic MapReduce algorithm matches each of the user's purchased and rated items to similar items then combines those similar items into a recommendation list.

We submitted the *ratings* provided by the 278 USA residents to Amazon Elastic MapReduce and calculated the average *cumulative* Euclidean distance between the lists ranked by the residents and the corresponding ones ranked by Amazon Elastic MapReduce, using Eq. 4. Figure 14 shows the results. We also asked the 32 UTA students to calculate the average *cumulative* Euclidean distance between the lists they ranked and the lists ranked by Amazon Elastic MapReduce, using Eq. 4.

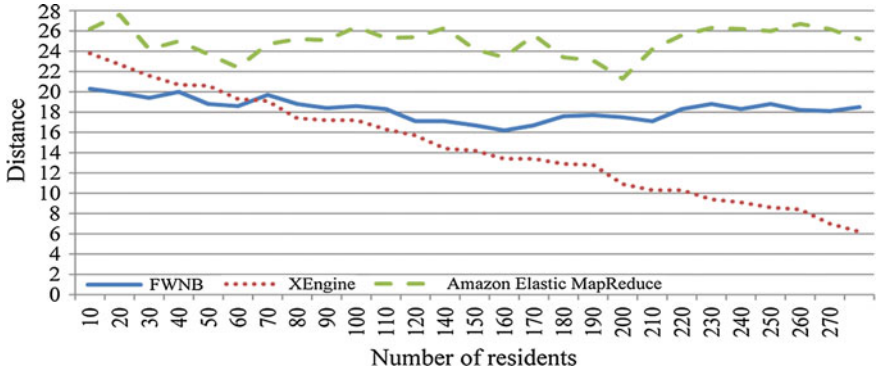


Fig. 14 Distances between the lists ranked by the residents and by Amazon Elastic MapReduce, FWNB, and XEngine

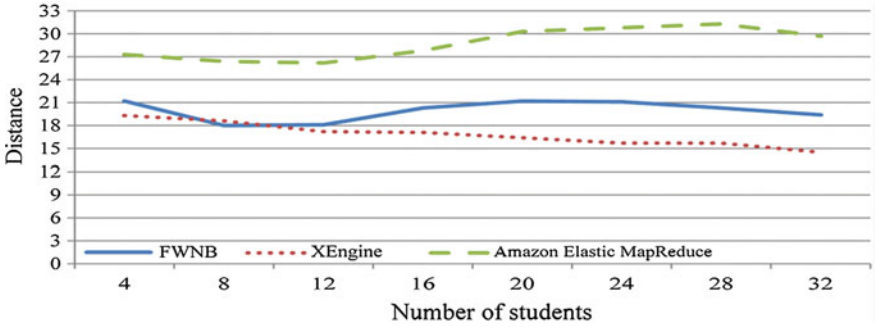


Fig. 15 Distances between the lists ranked by the students and by Amazon Elastic MapReduce, FWNB, and XEngine

Figure 15 shows the results. As Figs. 14 and 15 show, CF employing *group-based* profiling approach significantly outperformed the other approach. This is because a user usually rates only the *subset* of an item's features that reflects only *part* of his/her preferences. The techniques of CF employing *group-based* profiling may return items, whose some of their features are not rated by the user but are rated by the group, with which he/she shares interests.

9.7 Statistical Test of Significance

We aim at using z-test [28] to:

1. Determine whether the differences between individual Euclidean distances (*used in the evaluations presented in* Sects. 9.4 and 9.5) of each of the three systems are large enough to be statistically significant.
2. Test our hypothesis on specific Euclidean distances of the population mean.

Table 6 Average standard deviation D of the mean M

	XEngine	SPGProfile	FWNB
M	11.73	9.81	18.2
D	3.37	3.84	2.73

Table 7 Z-Score and the probability of a randomly selected list ranked by a system that achieved $D \leq X$

X	XEngine		SPGProfile		FWNB	
	Z score	$D \leq X$ (%)	Z score	$D \leq X$ (%)	Z score	$D \leq X$ (%)
20	2.45	98	2.65	99	0.66	75
16	1.27	89	1.61	94	−0.81	21
12	0.08	53	0.57	71	−2.27	0

The z-score is the distance between the sample mean and the population mean in units of the standard error. It is calculated as $Z=(X-M)/SE$ where X is the *mean sample*, M is the *population mean*, $SE=D/\text{sqrt}(n)$ is the *standard error of the mean* in which D is the *Average Standard Deviation of the mean*, and n is the *sample size*. For each system, Table 6 shows its mean (M) of its Euclidean distances, and its *Average Standard Deviation* (D) of the mean. As the values of D in Table 6 show: (1) the measured Euclidean distances of the three systems did not vary substantially with individual subjects' ratings, and (2) FWNB has the lowest D, because D is computed for FWNB based on the concept of biclusters while in the other two systems D is computed based on the concept of SDGs, and a bicluster is formed *solely* based on the *closeness* of its members' ratings.

Table 7 shows the z-scores for the Euclidean Distances of the three systems. Using the z-scores, we calculated the probability of a randomly selected subject (*from the 278 USA residents and 32 UTA student*), whose ranked list and a system's ranked list have an average Euclidean distance equal or less than a sample of mean (X). Column ($D \leq X$) in Table 7 shows the probabilities using a sample of three Euclidean mean (D). These probabilities were determined from a standard normal distribution table by using the z-scores as entries. For example, the probabilities for the systems to return a ranked list with Euclidean distance $\leq$16 (*see* Table 3) are as follow: XEngine: 89 %; SPGProfile: 94 %; and FWNB: 21 %. As the z-scores in Table 7 shows, the distances from the sample mean to the population mean are smaller for XEngine and SPGProfile. The table shows also that both XEngine and SPGProfile have a much higher probability for achieving Euclidean distance equal or less than the sample mean for a randomly selected subject's ranked list.

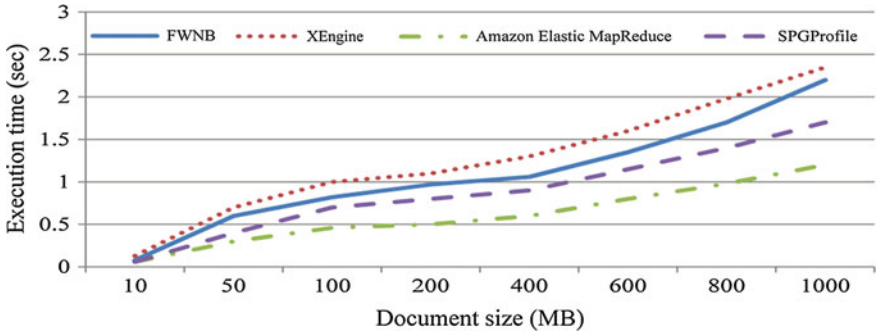


Fig. 16 Avg query execution time under variable document sizes

9.8 Search Efficiency Evaluation

We evaluated the *search efficiency* of XEngine by comparing its *average query execution time* with FWNB, Amazon Elastic MapReduce, and SPGProfile. We varied the size of the XML document 8 times in the range 10–1,000 MBs. We computed the *average query execution time* of each system under *each of the 8 document sizes*. Figure 16 shows the results. As the figure shows: (1) FWNB *slightly outperformed* XEngine, (2) the average query execution time of XEngine ranged from 1.1 to 1.3 times the execution time of an equivalent SPGProfile query, which is not expensive considering the *overhead* endured by XEngine because of adopting the *dynamic-implicit* techniques, and (3) the average query execution time of XEngine ranged from 1.2 to 1.9 times the execution time of an equivalent query of Amazon Elastic MapReduce (*recall that Amazon Elastic MapReduce employs user-based rather than group-based profiling*). Thus, the execution time of XEngine is not very expensive.

10 Conclusion

In this chapter, we proposed an XML-based CF recommender system, called XEngine. In the framework of XEngine, a social group is an entity that defines a group based on ethnic, cultural, religious, demographic, age, or other characteristics. We presented a novel approach to XML search that leverages group information to return most relevant query answers for users. User characteristics (e.g., social groups) are inferred *implicitly* by the system without involving the user. XEngine identifies social groups and their preferences *dynamically*. The preferences of a social group are determined from the preferences of its member users. XEngine outputs ranked lists of content items, taking into account not only the initial preferences of the user, but also the preferences of the user's various social groups.

We experimentally compared three approaches for *initializing* the preferences and ratings of a SDG. These are the *static initialization*, *dynamic initialization* using the XCDSearch approach [22], and *dynamic initialization* using the TopDown approach [24]. Based on the experiment results, we advocate employing the XCDSearch's approach for the sake of practicality and dynamicity. We experimentally evaluated the effectiveness of XEngine to dynamically *update* the ratings of a social group and to give ranked lists of items closer to those ranked by users. Towards this, we compared it with FWNP [21], using three different *data sets/methodologies*: MovieLens data set [17], test data attained from 278 subjects living in 20 US neighborhoods, and real-user evaluation. We found that XEngine outperforms FWNB. We experimentally evaluated the accuracy of the *dynamic-implicit* approach of group profiling (represented by XEngine) by comparing it with the *static-explicit* approach (represented by SPGProfile [23]). We found that the later approach *slightly outperforms* the former; thus, the *practicality and convenience* of the *dynamic-implicit* approach can *make up* for its slight lower accuracy. We experimentally found that CF approach employing *group-based* profiling techniques (represented by XEngine and FWNB) can outperform CF employing *user-based* profiling techniques (represented by Amazon Elastic MapReduce [2]). Finally, we evaluated the overhead endured by XEngine to employ its *dynamic-implicit* group-based profiling techniques. We found that the execution time of XEngine is not very expensive.

We will investigate in future work the expansion of the XEngine system to address the problem of opinion dynamics. We will investigate whether the process of repeatedly averaging the opinions of the members of a social group lead to a shift in the established and well known preferences of the social group. If so, we will investigate the factors involved in the shift to new preferences, such as the interaction of the social group with other social groups.

References

1. Aimeur E, Brassard G, Fernandez JM, Onana FS (2006) Privacy preserving demographic filtering. In: Proceedings of SAC'06
2. Amazon Elastic MapReduce (2012). http://awsmedia.s3.amazonaws.com/pdf/introduction-to-amazon-elastic-mapreduce.pdf
3. Amazon.com. http://www.amazon.com/grocery-breakfast-foods-snacks-ic/b?ie=UTF8&node=16310101
4. Bechhofer S, Harmelen F, Hendler J, Patel-Schneider P (2004) OWL web ontology language reference, W3C Recommendation
5. Breese J, Heckerman D, Kadie C (1998) Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of 14th UAI
6. Chamberlin D, Fankhauser P, Florescu D, Robie J (2007) XML query use cases, W3C Working Draft'07
7. Deshpande M, Karypis G (2004) Item-based top-n recommendation algorithms. ACM Trans Inf Syst 22(1):143–177
8. DiversityData.org and Harvard School of Public Health (2000) Exposure to neighborhood median income by race/ethnicity and income

9. FAQ Archives (2008) Religion and dietary practices. http://www.faqs.org/nutrition/PreSma/Religion-and-Dietary-Practices.html
10. Food culture and religion. http://www.betterhealth.vic.gov.au/bhcv2/bhcarticles.nsf/pages/food_culture_and_religion?opendocument
11. Gomes Jr MF, Canuto AM (2006) Carcara: a multi-agent system for web mining using adjustable user profile and dynamic grouping. In: Proceedings of IEEE/WIC/ACM IAT'06, Hong Kong, China
12. Herlocker J, Konstan JA, Riedl J (2004) Evaluating collaborative filtering recommender systems. ACM TOIS 22(1)
13. IMDB (Internet Movie Database) (2012). http://www.imdb.com/interfaces/#plain
14. Kittler P (1995) Food and culture in America: a nutrition handbook. West Publishing, Redding
15. Maponics (2012). Neighborhood to ZIP code correlation data. http://www.maponics.com/Neighborhood_ZIP_Codes/neighborhood_zip_codes.html
16. Minneapolis Census (2000) Selected economic characteristics by neighborhood, city of Minneapolis. http://www.ci.minneapolis.mn.us/citywork/planning/Census2000/maps/economic/
17. MovieLens (2012) Data set. http://www.grouplens.org/node/73
18. O'Connor P, Höpken W, Gretzel U (2008) Dynamic packaging using a cluster-based demographic filtering approach. In: The international conference in Innsbruck, Austria
19. Protégé ontology editor (2012). http://protege.stanford.edu/
20. Spink A, Jansen J, Ozmultu HC (2000) Use of query reformation and relevance feedback by excite users. Int Res Electron Networking Appl Policy 10(4):317–328
21. Symeonidis P, Nanopoulos A, Manolopoulos Y (2008) Providing Justifications in recommender systems. IEEE Trans Syst Man Cybern 38(6)
22. Taha K, Elmasri R (2010) XCDSearch: an XML Context-Driven Search Engine. IEEE Trans Knowl Data Eng (TKDE) 22(12):1781–1796
23. Taha K, Elmasri R (2010) SPGProfile: speak group profile. Inf Syst (IS) 35(7):774–790. Elsevier, Amsterdam
24. Tang L, Liu H, Zhang J, Agarwal N, Salerno J (2008) Topic taxonomy adaptation for group profiling. ACM Trans Knowl Discov Data 1(4)
25. Tesoro E (2001) Religious determinants of food choices
26. US Census Bureau (2006) USA Counties
27. Wang B, Bodily J, Gupta S (2004) Supporting persistent social groups in ubiquitous computing environments using context-aware ephemeral group service. IEEE PerCom
28. Warner R (2007) Applied statistics: from bivariate through multivariate techniques. Sage Publications, New York
29. Welcome to Food, Culture and Tradition. http://www.food-links.com/

Size, Diversity and Components in the Network Around an Entrepreneur: Shaped by Culture and Shaping Embeddedness of Firm Relations

Maryam Cheraghi and Thomas Schott

Abstract The network around an entrepreneur is conceptualized as having structural properties of size, diversity and a configuration of components. The Global Entrepreneurship Monitor has surveyed 61 countries with 88,562 entrepreneurs who reported networking with advisors. Cluster analysis of their relations revealed five components: a private network of advice relations with spouse, parents, other family and friends; a work-place network of boss, coworkers, starters and mentors; a professional network of accountants, lawyers, banks, investors, counselors and researchers; a market network of competitors, collaborators, suppliers and customers; and an international network of advice relations with persons abroad and persons who have come from abroad. Entrepreneurs' networking is unfolding in a culture of traditionalism versus secular-rationalism. Traditionalism is hypothesized to reduce diversity and size of networks and specifically reduce networking in the public sphere, but to enhance networking in the private sphere. Cultural effects on networking are tested as macro-to-micro effects on networking in two-level mixed linear models with fixed effects of traditionalism and individual-level variables and random effects of country. We find that traditionalism reduces diversity and overall networking and specifically networking in the work-place, professions, market and internationally, but enhances private networking. These cultural effects are larger than effects of attributes of the entrepreneur. The personal network around the entrepreneur provides an embedding of the business relations around the entrepreneurs' firm which are especially facilitated by the entrepreneur's networks in the public sphere.

Keywords Entrepreneurs · Networks · Culture · Embeddedness · Global Entrepreneurship Monitor

M. Cheraghi
University of Southern Denmark, Kolding, Denmark
e-mail: mche@sam.sdu.dk

T. Schott (✉)
University of Southern Denmark, Kolding, Denmark
e-mail: tsc@sam.sdu.dk

F. Can et al. (eds.), *State of the Art Applications of Social Network Analysis*, Lecture Notes in Social Networks, DOI: 10.1007/978-3-319-05912-9_16,

1 Introduction: Networks Around Entrepreneurs and Firms

A person's networking unfolds in the culture. In a traditional culture a person is in every-day life networking extensively with the family, typically, whereas in a secular-rationalistic culture a person networks more with professionals [5]. Culture may also affect the network around an entrepreneur [12]. An entrepreneur's networking is especially interesting for two reasons. First, entrepreneurs face considerably uncertainty in their endeavors and therefore are likely to depend heavily on their networking for resources [11]. Second, entrepreneurs typically assemble a wide spectrum of resources, including many kinds of knowledge, so their networks may be especially diverse and bridge many environments [8]. Indeed, the network is the entrepreneur's social capital from which the entrepreneur may benefit in the form of competitive advantage compared to those entrepreneurs whose networks provide fewer benefits [1]. These are good reasons for here analyzing entrepreneurs' networking with advisors in several environments.

An entrepreneur's personal networking is consequential for the entrepreneur's enterprise. The firm has business relations around it. The entrepreneur's personal network expectedly facilitates and promotes the business network around the entrepreneur's firm. This embedding of the business network in the personal network is a proposition of embeddedness [7].

Our study examines the causal scheme of culture, personal networking and business networking, where attributes of the entrepreneur and the firm are included as controls, Fig. 1.

We first conceptualize properties of the personal network around the entrepreneur, then specify hypotheses about cultural effects on the personal network, test them, and then turn to hypothesizing and testing subsequent effects on the business network around the entrepreneur's firm.

2 The Network Around an Entrepreneur: Size, Diversity and Components

The network around an entrepreneur has structural properties such as size, composition and diversity. Size of the network around an entrepreneur refers to the entrepreneur's number of advisors or number of different kinds of advisors, as a more qualitative range. Composition of the network around an entrepreneur refers to the networking with distinct environments and the prominence of each environment in the network. Diversity of the network around an entrepreneur refers to the spread of relations across many or even all of the environments [19].

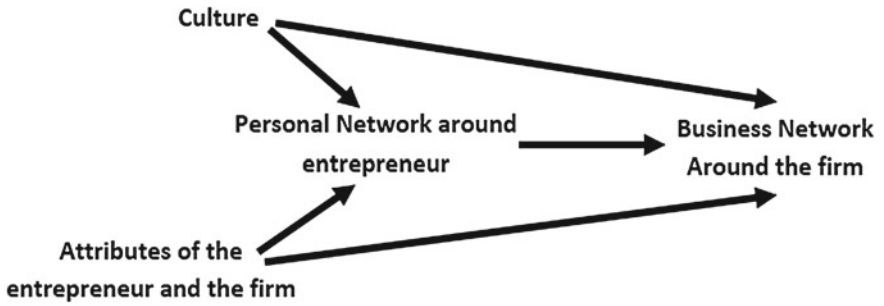


Fig. 1 Hypothesized effects among culture, personal networking and business networking

3 Cultures of Traditionalism and Secular-Rationalism

Culture shapes the networks around people. Networking differs between traditional culture and secular-rational culture [9]. A society may be pervasively secular-rational, or traditional culture may prevail, or its culture may be a blend on the continuum between secular-rational and traditional. Traditional culture and secular-rational culture are conceptualized and contrasted as pure types.

The traditional culture denotes the type of culture in which tradition is the authority guiding social life. The authority of tradition is localized in the family and religious leaders, and is exercised in youngsters' upbringing and when people are listening and seeking advice from these authorities. The contrasting type of culture is the secular-rational culture in which life is guided by considerations of ends and means toward ends, where benefits relative to costs are calculated and efficiency is pursued, and in this sense is rational (as in the concept secular-rational culture; [9, 26]). In the secular-rational culture authority is granted to expertise in ascertaining costs and benefits, especially based on science, and the expertise is institutionalized in the professions that teach youth and guide adults in pursuit of individual well-being and advantages.

In the traditional culture, networking is particularistic in the way that relations are based on prior ascribed ties, such as within the family and within a religious community, where internal trust and solidarity are high, whereas trust and solidarity are low to the outside, so external networking is sparse [15]. By contrast, in the secular-rational culture, networking is not particularistic but universalistic in the way that relations are formed more independently of prior ties, but formed more affectively by attraction between similar people, homophily, or more instrumentally by exchanges pursued for benefit, e.g. when seeking advice and buying expertise from professionals. In the traditional culture, networking is diffuse in the way that the relationship in a dyad typically is multi-stranded, a bundle of many kinds of relations, analytically distinct but intertwined in the dyad. By contrast, in the secular-rational culture, networking is not diffuse but specific in the way that the relationship

in a dyad typically is uniplex, with a specific content and pursued for that specific purpose. This traditional versus secular-rational dimension of culture shapes social life around the world [9] and specifically shapes networking [4, 5, 27].

4 Hypotheses About Cultural Effects on the Network Around an Entrepreneur

Our question is: how are size, diversity and components in the network around an entrepreneur shaped by culture? That traditional culture grants authority to family, whereas secular-rational culture grants more authority to professions, leads us to hypothesize that the entrepreneur's environment comprising family is more prominent in traditional culture than in secular-rational culture, and that the entrepreneur's environment comprising professions is more prominent in secular-rational culture than in traditional culture. Furthermore, the greater importance of family in traditional culture than in secular-rational culture induces the entrepreneur to pursue relations within the family rather than outside the family, which leads us to hypothesize that size of the entrepreneur's network decreases with traditionalism in the culture. The pursuit of relations outside the family entails a diversification of relations, so we also hypothesize that diversity in the entrepreneur's network increases with secular-rationalism and decreases with traditionalism.

4.1 Research Design

A study of culture affecting individual networking is a study of cultures at the macro-level and individuals at the micro-level. For studying culture, the unit of observation is a society, and the 'population' is all societies in the world. For studying networking, the unit of observation is an entrepreneur, and the 'population' is all entrepreneurs in the world, as two-level hierarchical populations, entrepreneurs nested within societies. A research design can use two-stage sampling, sampling societies and sampling entrepreneurs within each sampled society. When each stage of sampling is random or representative, inferential statistics can be used and results can be generalized to all entrepreneurs in all societies in the world. Observations comprise measures of culture in each sampled society and measures of networking of each sampled entrepreneur, and also measures of variables to be controlled for. A suitable technique for analyzing data on networks as they are affected by culture and control variables is two-level linear mixed modeling, with fixed effects of culture and control variables and random effects of country, also taking into account autocorrelation within each country. Coefficients in such models are then used for testing hypotheses about effects of culture on networking.

4.2 Data on Societies and Entrepreneurs

In our consortium Global Entrepreneurship Monitor we have in surveys 2009–2012 collected complete data on 88,562 networks around entrepreneurs in 61 societies, namely Algeria, Angola, Arab Emirates, Argentina, Australia, Bangladesh, Barbados, Bolivia, Bosnia and Herzegovina, Botswana, Brazil, China, Colombia, Costa Rica, Croatia, Czech Republic, Denmark, Ecuador, Egypt, El Salvador, Estonia, Ethiopia, Ghana, Greece, Guatemala, Hungary, Iran, Ireland, Israel, Jamaica, Jordan, Latvia, Lebanon, Malaysia, Mexico, Morocco, Namibia, Nigeria, Pakistan, Palestine, Peru, Poland, Portugal, Romania, Saudi Arabia, Singapore, South Africa, South Korea, Sweden, Syria, Taiwan, Thailand, Tonga, Trinidad and Tobago, Tunisia, Turkey, United States, Uruguay, Venezuela, Yemen and Zambia (www.gemconsortium.org). These 61 societies are fairly representative of the societies in the world, and the entrepreneurs are those identified in fairly random samples of the adults in each country [17]. Therefore the sampling is suitable for inferential statistics with generalization to all entrepreneurs in all societies [2, 6].

Each entrepreneur was asked about advice from various others [1, 10, 20]. Our list of potential advisors was culled from the literature (e.g. [8]) and pretested in 2008 in Denmark, Brazil, Iran, Latvia and South Korea [14], a pretest that led us to drop a few potential advisors and add a few other potential advisors to our final list of 20 possible advisors, Fig. 2. Each entrepreneur in the starting phase was asked "*Various people may give advice on your new business. Have you received advice from* [each of the 20 possible advisors]?" Slightly different formulations were used when asking entrepreneurs in the intending and operating phases of enterprise (the questionnaires are available on the website www.gemconsortium.org).

Culture in the dimension of traditionalism contrasted secular-rationalism has been measured in the World Values Survey [9, 28]. Traditionalism of culture is imputed for a few missing cases in our study as a weighted average of the measures available from similar neighboring countries, and we then standardize the measures of traditionalism across the 61 countries, Table 1.

Traditional culture is especially strong in El Salvador and Ghana, while secular-rational culture prevails around Europe. This measure of culture, high for traditional culture and low for secular-rational culture, is used for estimating effects of culture upon networks around entrepreneurs.

4.3 Entrepreneurs' Networking with Advisors

Entrepreneurs use some kinds of advisors frequently, while other potential advisors are rarely used, Table 2 (based on 88,562 entrepreneurs).

Table 2 shows that more than half of the entrepreneurs receive advice from friends, while few of the entrepreneurs receive advice from a lawyer.

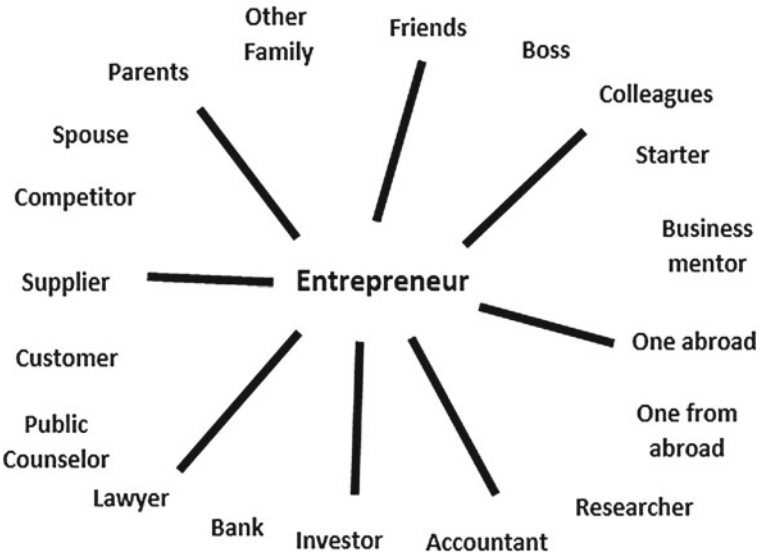


Fig. 2 An entrepreneur's network of advisors

Table 1 Countries with traditional culture and countries with secular-rational culture

Culture	Country	
Traditional culture	El Salvador	1.42
	Ghana	1.30
	Trinidad and Tobago	1.18
	Colombia	1.11
	Venezuela	1.04
	Egypt	0.98
	Nigeria	0.90
Secular-rational culture	Czech Republic	−1.97
	Denmark	−2.04
	Estonia	−2.07
	Sweden	−2.41

5 Size of the Network Around an Entrepreneur

The 20 relations tend to co-occur, their inter-correlations are all positive (Cronbach alpha is 0.85, and in an exploratory factor analysis the first eigenvalue is 5.3 and the second eigenvalue is below 1.7, and in a one-factor model all factor-loadings exceed 0.25). Thus, underlying the relations is an overall tendency toward networking. This makes it quite reasonable to use the concept of size of the network around an entrepreneur, and to operationalize size as the number of advisors used by the

Table 2 Entrepreneurs' use of advisors

Private network	Spouse	46 %
	Parents	44 %
	Other family	49 %
	Friends	55 %
Work-place network	Boss	10 %
	Co-workers	25 %
	Starter	20 %
	Business-mentor	30 %
Professional network	Lawyer	7 %
	Bank	7 %
	Accountant	12 %
	Public counselor	8 %
	Investor	10 %
	Researcher/inventor	7 %
Market network	Competitor	7 %
	Collaborator	11 %
	Supplier	16 %
	Customers	24 %
International network	Someone abroad	10 %
	Someone from abroad	10 %

Table 3 Countries with largest networks and countries with smallest networks around entrepreneurs

Arab Emirates	7.6
Tonga	6.7
Saudi Arabia	6.5
United States	6.5
Romania	6.4
…	…
Guatemala	2.7
China	2.8
Barbados	2.5
El Salvador	2.5
Tunisia	1.2

entrepreneur. Some entrepreneurs network more than others, and entrepreneurs in some countries have larger networks than entrepreneurs in other countries, Table 3.

Size of the networks around entrepreneurs in Arab Emirates exceeds 7 advisors, on average, while size of the networks around entrepreneurs in Tunisia more typically is just 1 advisor. Some of the variation among entrepreneurs in size of network can be attributed to their society (9 % as the eta-square estimated by anova).

We hypothesized that size of network is affected by culture in the way that traditionalism reduces networking. This hypothesis can be tested by the standardized coefficient in a model of network size (standardized) affected by national level of traditionalism (standardized). We include several control variables, here the individual characteristics of gender (coded 0 for males and 1 for females, then standardized),

Table 4 Size of the entrepreneurs' networks affected by traditionalism

Variable	Coefficient	Probability-value
Traditionalism	−0.13	0.003 one-sided
Gender female	−0.04	0.0001 two-sided
Age	−0.07	0.0001 two-sided
Education	0.09	0.0001 two-sided
Income	0.05	0.0001 two-sided
Household size	0.03	0.0001 two-sided
Self-efficacy	0.05	0.0001 two-sided
Opportunity-perception	0.08	0.0001 two-sided
Risk-willingness	−0.03	0.0001 two-sided
Phase intending	−0.07	0.0001 two-sided
Phase operating	−0.04	0.0001 two-sided
Algeria	−0.12	0.07 two-sided
Angola	0.56	0.0001 two-sided
Arab Emirates	1.02	0.0001 two-sided
Argentina	−0.08	0.11 two-sided
etc. for each country		

age (logarithm of years, then standardized), education (years, then standardized), household income (scaled 1, 2, 3 for thirds in sample of all adults within each country, then standardized), household size (logged, then standardized), entrepreneurial self-efficacy (coded 0 for unskilled and 1 for self-efficacious, then standardized), opportunity-perception (coded 0 for unaware and 1 for perceiving opportunities, then standardized), risk-willingness (coded 0 for fearing failure and 1 for risk-willing, then standardized). The three phases—intending, starting and operating—form a variable for intending contrasted to starting and another variable for operating contrasted starting (both as dummies, then standardized). The model is a hierarchical two-level mixed linear model, of entrepreneurs nested within countries, with fixed effects for traditionalism and the control variables, and random effects for country, within which also autocorrelation is taken into account, Table 4 (based on 61 countries with 64,824 entrepreneurs).

The effect of traditionalism is significant and negative, as we had hypothesized. The effect of traditionalism is stronger than the effect of education or any other measured attribute of the entrepreneur, and is the major effect upon the entrepreneurs' tendency to network.

In passing, we may also note that networking is greater for males than females, declines with age, increases with education, income, household size, self-efficacy and opportunity-perception, but declines a little with risk-willingness. More detailed analyses show that effects of attributes such as gender and education, however, differ between traditional culture and secular-rational culture [18]. Networking is less in the intending phase and in the operating phases than in the starting phase. We also note that there is a unique effect of each country upon size of network (shown for only a few of the 61 countries).

Looking beyond an entrepreneur's tendency toward networking or size of the network, and looking in greater detail, we can see that the network is differentiated into several components, networks in distinct environments.

6 Distinguishing Network Components

The composition of the network around an entrepreneur can be described as components or rather distinct networks in several environments [3]. A component thus denotes a cluster of co-occurring relations, advisors who tend to be used by the same entrepreneurs [19]. The composition in terms of network components can thus be revealed by a cluster analysis. Here we use the default variable clustering procedure in SAS (an exploratory factor analysis identifies the same clusters). We discerned five network components among the 20 advisors:

- Private network of relations of receiving business advice from spouse, parents, other family and friends;
- Work-place network of relations with boss, co-workers, starters and more experienced business-people;
- Professional network of relations with lawyer, bank, accountant, public business counselor, potential investor and researcher or inventor;
- Market network of relations of receiving advice from competitor, collaborator, supplier and customers; and
- International network of relations with someone who is abroad and someone who has come from abroad.

Diversity of the network around an entrepreneur refers to networking widely across the five environments. Diversity of an entrepreneur's network may be operationalized as the number of components that the entrepreneur receives advice from.

A component in the network around an entrepreneur is more or less prominent. For example, the entrepreneur's private network may be highly prominent, more utilized than her work-place network, her professional network, her market network and her international network. For another entrepreneur the private network may be less prominent or less salient than his other network components. Each component in the network around the entrepreneur thus has a degree of prominence. Prominence of an entrepreneur's private network can be measured as the number of advisors who are drawn from the private environment, i.e. the size of the private network. For an entrepreneur, the sum of the five prominences—across the five environments—is thus the size of the whole network around the entrepreneur. An alternative way of measuring prominence of the private network is by a relative measure, as the proportion of the advisors in the whole network who are drawn from the private environment. For an entrepreneur, the relative prominences of the five components are thus five proportions summing to 1. In this article we measure prominence of a component by the count of advisors within that environment (substantively similar results are obtained by measuring prominence relative to network size; [22]).

7 Hypotheses About Specific Environments and Diversity

Having distinguished five network components we can now use our initial considerations of culture to specify our hypotheses about each environment and also about diversity:

- The private network is especially prominent in cultures that are traditional.
- The work-place network is little prominent in cultures that are traditional.
- The professional network is little prominent in cultures that are traditional.
- The market network is little prominent in cultures that are traditional.
- The international network is little prominent in cultures that are traditional.
- Diversity of the network around an entrepreneur is low in cultures that are traditional.

These six specific hypotheses can now be tested.

7.1 Prominence of the Private Network

Prominence of the private network, as the count of advisors in the private environment, varies among entrepreneurs. This variation in prominence of the private networks around entrepreneurs is partly a variation among societies, Table 5. Entrepreneurs in Tonga, Saudi Arabia, Zambia and other countries in Southern Asia and Africa have networks in which the private component is especially prominent. Conversely, the private network has low prominence for entrepreneurs in several countries in Europe. Much of the variation among entrepreneurs in prominence of the private network can be attributed to their society (9 % as the eta-square estimated by anova).

Prominence of the private network was hypothesized to be shaped by culture in the way that prominence is higher in traditional culture than in a secular-rational culture. This hypothesis can be tested by the standardized coefficient in a model of prominence of the private network affected national level of traditionalism (a hierarchical two-level mixed linear model with controls as in the above analysis of effects upon network size), Table 6 (based on 61 countries with 64,824 entrepreneurs).

The culture of traditionalism versus secular-rationalism has a significant and positive effect, as was hypothesized. The effect is quite strong, and stronger than any effect of an attribute of the entrepreneurs.

7.2 Prominence of the Work-Place Network

Prominence of the work-place network, as number of advisors in the work-place environment, also varies among entrepreneurs, and some of the variation in promi-

Table 5 Countries with high prominence and countries with low prominence of private networks

Tonga	2.9
Saudi Arabia	2.7
Zambia	2.7
Pakistan	2.6
Nigeria	2.6
…	…
Sweden	1.3
Uruguay	1.2
S. Korea	1.2
Poland	1.2
Tunisia	1.0

Table 6 Prominence of the private network affected by traditionalism

Variable	Coefficient	Probability-value
Traditionalism	0.11	0.003 one-sided
etc. as in Table 4		

Table 7 Countries with high prominence and countries with low prominence of the work-place network

Arab Emirates	1.7
United States	1.6
Syria	1.6
Estonia	1.5
Saudi Arabia	1.5
…	…
China	0.4
Ethiopia	0.4
Ghana	0.4
El Salvador	0.4
Tunisia	0.1

nence of work-place network can be attributed to society (8 % is the eta-square estimated by anova). The work-place network is especially prominent in Arab Emirates and United States and has especially low prominence in Tunisia and El Salvador, Table 7.

Prominence of the work-place network was hypothesized to be shaped by culture in the way that prominence is lower in traditional culture than in secular-rational culture. This hypothesis is tested by the standardized coefficient in a hierarchical mixed linear model of prominence of the work-place network affected by national level of traditionalism, Table 8 (based on 61 countries with 64,824 entrepreneurs).

The culture of traditionalism has a significant effect in the way that traditionalism reduces prominence of the work-place network, as we had hypothesized. Traditionalism has a strong effect, stronger than the effect of any attribute of the entrepreneur.

Table 8 Prominence of work-place network affected by traditionalism

Variable	Coefficient	Probability-value
Traditionalism	−0.16	0.0001 one-sided
etc. as in Table 4		

Table 9 Countries with high prominence and countries with low prominence of professional networks

Arab Emirates	1.5
United States	1.5
Romania	1.5
Ireland	1.3
Latvia	1.3
…	…
Guatemala	0.10
Ethiopia	0.10
Bangladesh	0.07
China	0.07
Tunisia	0.06

Table 10 Prominence of professional network affected by traditionalism

Variable	Coefficient	Probability-value
Traditionalism	−0.17	0.0001 one-sided
etc. as in Table 4		

7.3 Prominence of the Professional Network

Prominence of the professional network also varies among entrepreneurs, Table 9, and some of the variation in prominence of professional network can be attributed to society (9 % as the eta-square estimated by anova). Professional networks are especially prominent in Arab Emirates, United States and Romania, but have especially low prominence in Tunisia, China and Bangladesh.

Prominence of the professional network was hypothesized to be shaped by culture in the way that prominence is low in traditional culture and high in secular-rational culture. This hypothesis is tested by the standardized coefficient in a model of prominence of the professional network affected by traditionalism, Table 10 (based on 61 countries with 64,824 entrepreneurs).

The culture of traditionalism has a significant effect in the way that traditionalism reduces prominence of the professional network, as we had hypothesized. Traditionalism has an effect that is stronger than the effect of any attribute of the entrepreneur.

7.4 Prominence of the Market Network

Prominence of the market network also varies among entrepreneurs, and some of the variation in prominence of market network can be attributed to society (9 % as

Table 11 Countries with high prominence and countries with low prominence of the market networks

Arab Emirates	1.4
United States	1.4
Sweden	1.2
Estonia	1.2
Australia	1.2
…	…
Bangladesh	0.2
Bolivia	0.2
China	0.2
Guatemala	0.1
Tunisia	0.03

Table 12 Prominence of market networks affected by traditionalism

Variable	Coefficient	Probability-value
Traditionalism	−0.19	<0.00003 one-sided
etc. as in Table 4		

the eta-square estimated by anova). Market networks are especially prominent in the Arab Emirates and United States, but have especially low prominence in Tunisia and Guatemala, Table 11.

Prominence of the market network was hypothesized to be shaped by culture in the way that prominence is low in traditional culture and high in secular-rational culture. This hypothesis is tested by the standardized coefficient in a model of prominence of the market network affected by traditionalism, Table 12 (based on 61 countries with 64,824 entrepreneurs).

The culture of traditionalism has an effect on market networking in the way that traditionalism reduces prominence of the market network, as was hypothesized. The effect is large, larger than the effect of any attribute of the entrepreneur.

7.5 Prominence of the International Network

Prominence of international networking varies among entrepreneurs, and some of the variation in this prominence is attributed to society (6 % as eta-square estimated by anova). The international network is especially prominent in the small Arab Emirates and Tonga and has especially low prominence in the huge nation China, Table 13.

Prominence of the international network was hypothesized to be shaped by culture in the way that prominence is low in traditional culture and high in secular-rational culture. This hypothesis is tested by the standardized coefficient in a model of prominence of the international network affected by traditionalism, Table 14 (based on 61 countries with 64,824 entrepreneurs).

The culture of traditionalism has an effect on international networking in the way that traditionalism reduces prominence of the international network, as was

Table 13 Countries with high prominence and low prominence of international networks

Arab Emirates	0.7
Tonga	0.6
Romania	0.5
Estonia	0.5
Singapore	0.5
…	…
Iran	0.06
Ghana	0.06
Brazil	0.04
Tunisia	0.02
China	0.01

Table 14 Prominence of international networks affected by traditionalism

Variable	Coefficient	Probability-value
Traditionalism	−0.08	0.02 one-sided
etc. as in Table 4		

Table 15 Countries with high diversity and countries with low diversity of networks

Arab Emirates	3.3
Estonia	3.1
United States	3.1
Syria	3.0
Australia	3.0
…	…
Ghana	1.4
Ethiopia	1.4
China	1.4
Guatemala	1.3
Tunisia	0.7

hypothesized. The effect is substantial and larger than the effect of any of the attributes of entrepreneurs.

7.6 Diversity of the Network Around an Entrepreneur

Diversity of the network varies among entrepreneurs, and some of the variation in this diversity is attributed to society (12% as eta-square estimated by anova). The diversity is especially high in the Arab Emirates and is especially low in Tunisia, Table 15.

Diversity of the network was hypothesized to be shaped by culture in the way that diversity is low in traditional culture and high in secular-rational culture. This hypoth-

Table 16 Diversity of networks affected by traditionalism

Variable	Coefficient	Probability-value
Traditionalism	−0.19	<0.0003 one-sided
etc. as in Table 4		

esis is tested by the standardized coefficient in a model of diversity of network affected by traditionalism, Table 16 (based on 61 countries with 64,824 entrepreneurs).

The culture of traditionalism has an effect on diversity in the way that traditionalism reduces diversity of networks, as was hypothesized. The effect is large, larger than the effect of any of the attributes of entrepreneurs.

8 A Consequence of an Entrepreneur's Networking: Embeddedness of the Firm's Business Relations

The proposition of embeddedness argued by Granovetter [7] posits that, typically, a business relation is not a pure exchange, which is short-term, contractual and uncommitted, but is facilitated by a personal bond, which is long-term, trusting and committed. The concept of embeddedness denotes the overlap and intertwining between social bonds and economic exchanges within and between organizations. Network theory argues that embeddedness promotes actors' motivations from the narrow pursuit of immediate economic gains toward the enrichment of bonds through trust and reciprocity [16, 24]. The resulting trust reduces transactional uncertainty and creates opportunities for the exchange of goods and services which are difficult to price or enforce contractually [24]. Therefore, social relations facilitate business agreements by improving trust and morality. In spite of the fact that institutional contracts have been recognized to be essential between partners, such arrangements alone cannot ensure commitment and no fraud, and usually are too costly. The presence of trust reduces cheating in the business relations, mostly because the most basic motivation to abstain from cheating on business agreements is to keep reputation either within the market or among the partners' network. In practice, people more contend to have business arrangements with those they trust, and the main source of this trust stems from better information and moral obligation anchored on bonds. Better information includes information which comes from a trusted informant, someone who has already worked with that firm, and found it to be reliable. This informant could be either inside or outside their firm. In addition, firms are cautious in keeping their good reputation, because they tend to benefit from the advantages of repeating business contracts with the old partners. This kind of information is cheap, more detailed and more accurate, trustworthy, and beyond pure economic incentives [7].

Embeddedness is here examined in terms of how the network of advisors around an entrepreneur facilitates the business network around the entrepreneur's firm. We hypothesize that the more the entrepreneur is networking, the stronger will be the business relations around the entrepreneur's firm. More specifically, we hypothesize

Table 17 Countries with dense business relations and countries with sparse business relations

Tunisia	0.6
Croatia	0.4
Estonia	0.4
Palestine	0.4
Poland	0.4
…	…
Zambia	0.1
Botswana	0.1
Namibia	0.1
El Salvador	0.1
Ghana	0.1

that the business relations are facilitated more by work-place, professional, market and international networking than by private networking.

In the Global Entrepreneurship Monitor survey in 2012 we measured business relations around 6,543 firms in 33 countries. Those owning-managing an established firm were asked seven questions, *I will now ask how your new business works with other enterprises and private and public organizations. Is your business working together with other enterprises or organizations … to produce goods or services? … to procure supplies? … to sell your products or services to your current customers? … to sell your products or services to new customers? … to create new products or services to your current customers? … to create new products or services to new customers? … to make your business more effective?* For each of these seven kinds of business relations, the strength of the relation was measured on a scale from 0 to 1 (0 for no collaboration, .5 for collaboration that is not so intense, and 1 for collaboration that is intense). The firm's business networking can thus be measured by the mean strength of the seven relations. This measure of a firm's business networking, on the scale from 0 to 1, is suitable for analyses of how the firm's business networking is embedded in the personal network around the entrepreneur.

Business networking varies among entrepreneurs' firms, and some of this variation is attributed to society (13 % as eta-square estimated by anova). Firm networking is especially high in Tunisia, Croatia and Estonia, and is especially low in countries in Sub-Saharan Africa, Table 17.

Firm networking was hypothesized to be embedded in and facilitated by the personal advice network around the entrepreneur. Specifically, we hypothesize that firm networking is promoted by the entrepreneur's work-place network, professional network, market network and international network, more than by the private network. Moreover, firm networking may well be shaped by culture in the way that firm networking is lower in traditional culture than in secular-rational culture, because much of the firm networking is collaboration around innovating. These hypotheses are tested by the standardized coefficients in a model of firm networking affected by traditionalism and by the five component networks, controlling for attributes of the entrepreneur and the firm, Table 18 (based on 33 countries with 6,543 entrepreneurs).

Table 18 Firms' relations (mean; standardized) affected by traditionalism and components around entrepreneur

	Coefficient	Probability-value
Traditionalism (standardized)	−0.08	0.10 one-sided
Private network (scale 0 to 4; standardized)	0.03	0.01 one-sided
Work-place network (scale 0 to 4; standardized)	0.11	0.0001 one-sided
Professional network (scale 0 to 6; standardized)	0.07	<0.00003 one-sided
Market network (scale 0 to 4; standardized)	0.13	0.0001 one-sided
International network (scale 0 to 2; standardized)	0.06	0.30 one-sided
Diversity (scale 0 to 5; standardized)	0.01	0.30 one-sided
Gender (0 male, 1 female; standardized)	−0.02	0.03 two-sided
Age (log of years; standardized)	0.01	0.52 two-sided
Education (standardized within countries; then all standardized)	0.05	<0.00005 two-sided
Self-efficacy (1 self-efficacious, 0 not; standardized)	0.01	0.29 two-sided
Opportunity-perception (1 perceiving, 0 not; standardized)	0.03	0.01 two-sided
Risk-willingness (1 willing, 0 averse; standardized)	0.01	0.46 two-sided
Motivation: opportunity (1 opportunity, 0 not; standardized)	−0.01	0.53 two-sided
Motivation: need for income (1 need, 0 not; standardized)	−0.07	<0.00005 two-sided
Motivation: mix of the two (1 mix, 0 not; standardized)	−0.03	0.003 two-sided
Motivation: other (1 other, 0 not; standardized)	−0.01	0.59 two-sided
Sole proprietorship (1 sole, 0 shared; standardized)	−0.05	0.35 two-sided
Owners (log of number of owners; standardized)	0.08	0.08 two-sided
Firm-age (log of years; standardized)	−0.01	0.29 two-sided
Firm-size (log of persons; standardized)	0.21	<0.00005 two-sided
Intercept	0.08	
Algeria	0.01	0.99 two-sided
Argentina	−0.08	0.33 two-sided
etc. for each country		

Traditionalism reduces firms' business relations, as was expected. Each of the five component networks appears to enhance business relations, but not equally. The firm's business relations are facilitated and promoted by the entrepreneur's work-place, professional, market and international networking. But the private network apparently has no substantial effect.

In short, the firm's business relations are embedded in and facilitated by the personal network around the entrepreneurs, especially by the market network and the work-place network, and to lesser degree also by the professional network and the international network, but not notably by the private network.

9 Conclusions

The network around an entrepreneur has here been conceptualized as having the structural properties of size, diversity and composition as network components, each with its own prominence.

Networking is embedded in culture with the dimension of traditionalism versus secular-rationalism. Cultural effects were tested on the networks around 88,562 entrepreneurs in 61 societies, where each entrepreneur reported relations with up to 20 advisors.

We hypothesized seven effects of the conditions of traditionalism upon the six properties of the network around an entrepreneur,

- traditionalism decreases size of the network around an entrepreneur,
- traditionalism decreases diversity of networking, and specifically for the five components,
- traditionalism increases private networking,
- traditionalism decreases work-place networking,
- traditionalism decreases professional networking,
- traditionalism decreases market networking,
- traditionalism decreases international networking,

All seven effects were found to be significant as hypothesized, and actually quite large, larger than effects of individual attributes.

An entrepreneur's personal networking for advice was thought to be consequential, in the way that it entails an embeddedness of the business relations of the entrepreneur's firm. We hypothesized that

– size of the entrepreneur's personal network increases the firm's business relations,
– The entrepreneur's work-place, professional, market and international networking increases the firm's business relations more than the entrepreneur's private networking does.

These effects were found to be significant as hypothesized.

Finally, we also hypothesized that traditionalism reduces business networking. Also this hypothesis was supported.

These effects are summarized in Fig. 3. The positive effects are solid arrows and the negative effects are dotted arrows. Thickness of an arrow represents the magnitude of the effect as indicated by the standardized coefficient.

The networks around entrepreneurs are shaped not only by the culture of their societies, but also by their personal characteristics, e.g. gender, age and education, and by characteristics of their enterprises, e.g. phase and ownership. These attributes have effects on networking which are not as strong as the effects of culture.

The entrepreneurs' personal networks have consequences for their performance, specifically their innovation and expectations for growth of their firms [21, 23]. Likewise, the firms' business networks have consequences for their performance, specifically their innovation, exporting and growth-expectations [13, 25].

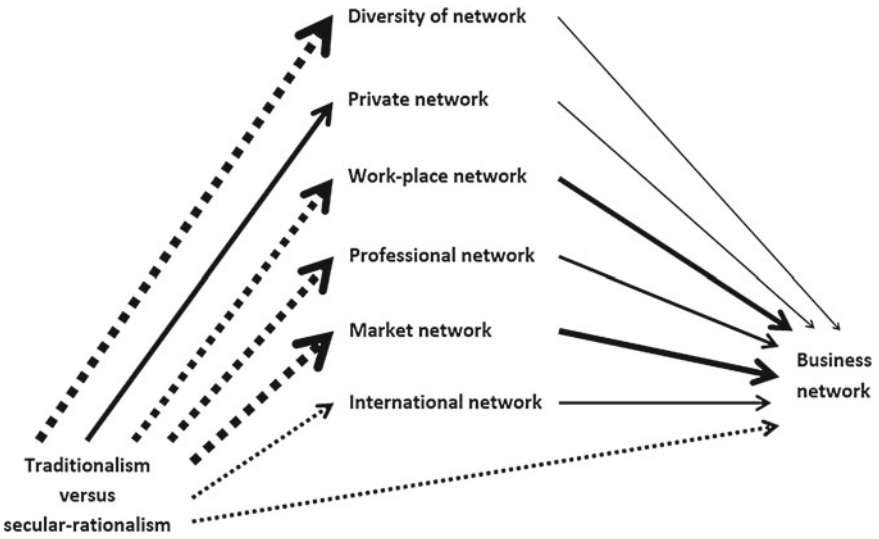


Fig. 3 Effects among culture, the entrepreneur's personal network and the firm's business network

Acknowledgments This study relies mainly on data collected in the Global Entrepreneurship Monitor program, but sole responsibility for their analysis and interpretation rests with the authors.

References

1. Bahn S, Greco S, Farsi JY, Rastrigina O, Schott T (2011) Entrepreneurs' expected returns affected by their networks: a cross-national study using Global Entrepreneurship Monitor data. Asia Pac J Innov Entrepreneurship 5(2):75–96
2. Bosma N, Coduras A, Litovski Y, Seaman J (2012) A report on the design, data and quality control of the Global Entrepreneurship Monitor. GEM Manual. www.gemconsortium.org
3. Burt RS, Schott T (1989) Relational contents in multiple network systems. In: Freeman LC, White DR, Romney AK (eds) Research methods in social network analysis. George Mason University Press, Fairfax, pp 185–213
4. Fischer CS (1982) To dwell among friends: personal networks in town and city. University of Chicago Press, Chicago
5. Freeman LC, Ruan D (1997) An international comparative study of interpersonal behavior and role relationships. L'Année Sociologique 47(1):89–115
6. Global Entrepreneurship Monitor (2013). http://www.gemconsortium.org
7. Granovetter M (1985) Economic action and social structure: the problem of embeddedness. Am J Sociol 91:481–510
8. Greve A, Salaff JW (2003) Social networks and entrepreneurship. Entrepreneurship: Theor Pract 28: 1–22
9. Inglehart R, Welzel C (2005) Modernization, cultural change and democracy. Cambridge University Press, New York
10. Kelley DJ, Brush CG, Green PG, Litovski Y (2011) Global Entrepreneurship Monitor 2010 women's report. GERA/GEM. http://www.gemconsortium.org

11. Klyver K, Hindle K (2007) The role of social networks at different stages of business. Small Bus Res 15:22–38
12. Klyver K, Hindle K, Meyer D (2008) Influence of social network structure on entrepreneurship participation—a study of 20 national cultures. Int Entrepreneurship Manage J 4:331–347
13. Lencher C, Dowling M (2003) Firm networks: external relationships as sources for the growth and competitiveness of entrepreneurial firms. Entrepreneurship Reg Dev Int J 15(1):1–26
14. Martínez AC, Levie J, Kelley DJ, Sæmundsson R, Schøtt T (2010) Global Entrepreneurship Monitor special report: a global perspective on entrepreneurship education and training. GERA/GEM. http://www.gemconsortium.org
15. Parsons T, Shils E (eds) (1951) Toward a general theory of action. Harvard University Press, Cambridge
16. Powell WW (1990) Neither market nor hierarchy: network forms of organization. In: Straw B, Cummings LL (eds.) Research in organizational behavior. JAI Press, Greenwich, CT, pp 295–336
17. Reynolds P, Bosma N, Autio E, Hunt S, de Bono N, Servais I, Lopez-Garcia P, Chin N (2005) Global Entrepreneurship Monitor: data collection design and implementation 1998–2003. Small Bus Econ 24:205–231
18. Schott T (2013) Entrepreneurs' networking in private and public spheres: shaped by culture, gender and university education (Under peer-review), (Submitted)
19. Schott T (2013) Components in the network around an actor. In: R Alhajj, J Rokne (eds) Encyclopedia of social network analysis and mining. Springer, Heidelberg (In press)
20. Schott T (2011) Training and network organization in entrepreneurship. University of Southern Denmark, Kolding
21. Schott T, Ashouri S (2012) Entrepreneurs' growth-expectations: shaped by culture and networking. In: Presented at RENT XXVI conference, Lyon
22. Schott T, Cheraghi M (2012) Entrepreneurs' networks: size, diversity and composition shaped by cultures of rationality and trust. In: Proceedings the 2012 IEEE/ACM international conference on advances in social network analysis and mining, Aug 2012, Istanbul, Turkey. IEEE Computer Society, Los Alamitos, California, pp 220–226, ISBN 978-0-7695-4799-2
23. Schott T, Sedaghat M (2012) Entrepreneurs' innovation shaped by their networking and by national education: a global study. In: Uddevalla symposium 2012: entrepreneurship and innovation networks. University West, Trollhättan, Sweden, pp 707–726
24. Uzzi B (1996) The sources and consequences of embeddedness for the economic performance of organizations: the network effect. Am Sociol Rev 61:674–698
25. Uzzi B (1999) Embeddedness in the making of financial capital: how social relations and networks benefit. Am Sociol Rev 64:481–505
26. Weber M (1920) Economy and society. University of California Press, Berkeley
27. Wellman B (1999) Networks in the global village. Westview Press, Boulder
28. World Values Survey. http://www.worldvaluessurvey.org/wvs/articles/folder_published/article_base_111

Content Mining of Microblogs

M. Özgür Cingiz and Banu Diri

Abstract Emergence of Web 2.0, internet users can share their contents with other users using social networks. In this chapter microbloggers' contents are evaluated with respect to how they reflect their categories. Migrobloggers' category information, which is one of the four categories that are economy sport, entertainment or technology, is taken from wefollow.com application. 3337 RSS news feeds, whose category labels are same with microbloggers' contributions, are used as training data for classification. Unlike the similar studies if a feature of microblog doesn't appear in RSS news feeds as a feature, this feature is omitted so abbreviations and nonsense words in microblogs can be eliminated. In this study two types of users' contributions are taken as test data. These users are normal microbloggers and bots. Classification results show that bots provide more categorical content than normal users.

1 Introduction

Recent advancements in Web 2.0, people can't be regarded as simple content reader they can also contribute content as writers. Web 2.0 introduces concepts like social network, blog and microblogs with internet users. Users share their opinions, feelings, images, favorite videos and other user's contributions as microblog content.

Microblogs differ from blogs. Microblogs have size limitation for content. Twitter is one of the most popular microblog applications because of its easy sign up process, easy to use and mobile access. It has limitation of 140 characters for content. User contribution is called as tweet in Twitter.

M. Ö. Cingiz (✉) · B. Diri
Computer Engineering Department, Yıldız Teknik Üniversitesi, Istanbul, Turkey
e-mail: mozgur@ce.yildiz.edu.tr

B. Diri
e-mail: banu@ce.yildiz.edu.tr

F. Can et al. (eds.), *State of the Art Applications of Social Network Analysis*,
Lecture Notes in Social Networks, DOI: 10.1007/978-3-319-05912-9_17,

In Twitter users can follow other users with respect to their field of interest. Followers expect users who are followed in terms of their categorical information, to share content about their field of interests. This study aims to evaluate two types of users' contents according to specifying they reflect their category or not.

Users can find out microblogger's category information with using some applications such as wefollow.com. Users can describe their field of interests and category which they provide tweet about. Because of the character limitations microbloggers' contents can consist of abbreviations and nonsense words so this decrease success of classification. In this chapter, we aim to eliminate this kind of features that lead to decrease success of classification.

Similar studies can be separated into two area. First one is to find out user who shares similar interest. Second is to obtain patterns from microblogs. Degirmencioglu [1] extracts word-hashtag, word-user and hashtag-user pairs from tweets to discover users' common interest areas. Yurtsever [2] classifies microbloggers according to their contents with using semantic resources. Akman [3] extract categorical features from 150 microbloggers' contents. Aslan [4] uses news pattern similarity for discovering microbloggers who broadcast news content. Pilavcilar [5] classify texts with using text mining techniques that some of them are used in this study. Güc [6] uses microbloggers' contents and text classification techniques to measure convenience of users' categories.

In this study in part two we examine data sets and their features. In third part analyzing prepared model and model steps to find out users whose contents are more valuable for its related category. In last part, we refer classification results and feature work.

2 Data Sets

This study consists of two parts. First part is training part and second is test part. Training part data consists of RSS news feeds. Content suppliers like BBC, CNN, SKYNews provide their subscribers news with RSS format. RSS is a kind of web feed format like atom. Users can follow news with using web browsers or aggregators. We use RSS4j java library for getting RSS news feeds. 3337 RSS news feeds whose category is one of the four categories which are economy sport, entertainment or technology. 924 entertainment, 738 technology, 721 economy and 954 sport RSS news feeds are taken for build training model.

In test part 10,630 tweets are obtained form 32 bot users and 30 normal users. Category information of bot users and normal users are taken from wefollow.com application. Category labels are the same with training case. We obtain users' tweets with using Twitter4j java library.

3 Proposed System

Figure 1 shows the steps of the proposed system. Proposed system consists of two phase. These phases are training and testing phases.

In training phase RSS news feeds are used for building training model. Content distributors supply categorical information of RSS news feeds so we can obtain category of training data. However, summaries of news also consist of valuable features so taking RSS news feeds as training data reduce feature of training data set. RSS news feeds and microbloggers' contents are taken in same time period for checking up-to-dateness of microblogers' contents.

After retrieval of RSS news feeds, RSS news feeds are processed for text classification. In text classification area vector space model is used as representing text documents as vectors in vector space. In training phase steps of text preprocessing, feature weighting, dimension reduction, specifying term count threshold are applied to RSS news feeds respectively. After preprocessing steps, Support Vector Machines and Multinominal Naive Bayes are used separately as classifier for training phase.

In test phase tweets of 30 normal microbloggers and tweets of 32 bots, which their categorical information is obtained from wefollow.com application, are used. Microbloggers' categories are sport, economy, entertainment and technology. Before the selection of features of microbloggers' contents, removing punctuations and tokenization steps are applied. Microbloggers' tweets split into their tokens (features, words). If any word that is part of microbloggers' tweet doesn't be in training feature set, this word is omitted. Features are only taken from training set and search these features in microbloggers' contents because of abbreviations and nonsense words in microblogs. If these words are regard as features, classification success rate is decreasing and testing phase results are specious. After feature specifying steps, features are weighted.

Contents of tweets can be hyperlink of picture or video so in testing phase we eliminate hyperlinks. After selection of only training features and removal of hyperlinks, some tweets become featureless. Featureless tweets are meaningless as test data so we specify 3 different term count threshold values. Tweets must include at least three, four or five words as test data. Testing is implemented with these 3 different threshold values. Test data is given to training model for classification that is formed by Support Vector Machines classifier and Multinominal Naive Bayes classifier. In this section we explain all training and testing steps clearly.

3.1 Preprocessing

Removal of punctuations, tokenization, and selection of features in terms of their linguistic information, stemming and elimination of stop words are preprocessing steps that are used in text mining area. According to selection of linguistic features only nouns and verbs are used as features.

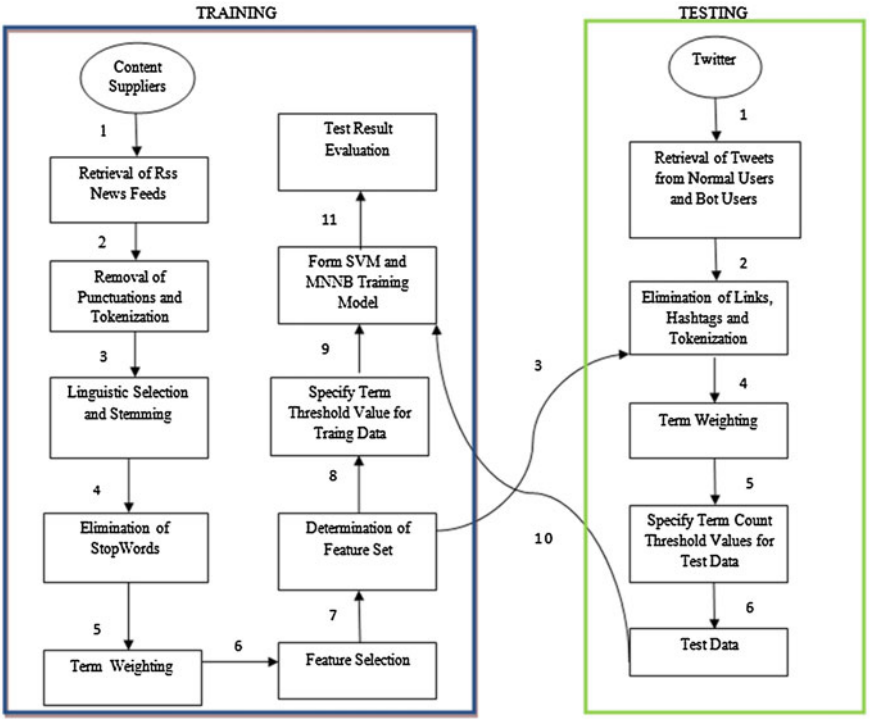


Fig. 1 Proposed system structure

3.1.1 Removal of Punctuations and Tokenization

First preprocessing step is removal of punctuations of RSS news feeds. After removal of punctuations, word tokenization is applied to train data. Word tokenization splits RSS news feeds into their words. Tokens also can be n-grams or collocations but in this study words are taken as features of text. In vector space RSS news feeds are shown as vectors, tokens of RSS news feeds are dimension of concerned vector.

3.1.2 Linguistic Selection and Stemming

In previous text classification works features are evaluated separately according to their linguistic labels. Classification results which are obtained using different features according to their linguistic information shows that nouns and verbs are more valuable features than other types [7, 8]. In this study only nouns and verbs are used as features. Pronouns, adjective, conjunctions are eliminated. Stanford[1] part-of-speech tagger is used for getting linguistic information of words.

[1] http://nlp.stanford.edu/software/tagger.shtml

Words can be in different formats in texts. Such as "*children, child*" is *different* forms of noun *child* or "*drink, drank, drunk*" is different forms of verb *drink*. Stemming is necessary for successful classification and feature reduction.

3.1.3 Elimination of Stop Words

Stop words are commonly used in every category so they don't have any differentiation impact. Stop words decrease classification success. We eliminate stop words from feature set in preprocessing phase.

3.2 Term Weighting

In vector space model a text document is symbolized as vectors, words (features, terms) in this text document are symbolized as dimensions of vector. In vector space model every word has a weight value if it is in text document. In this chapter, term frequency-inverse document frequency (tf-idf) is used for weighting for features. In related works show that selection of weighting approach is more important than selection of kernel function for Support Vector Machines classifier [9, 10].

In tf-idf weighting term frequency, tf, gives number of times a term occurs in a text document. Inverse document frequency, idf, gives number of times a term occurs in whole text documents. If any terms occur in every document, it is worthless feature for classification. Valuable features for classification have high term frequency score and low inverse document score. Equations 1 and 2 show calculation of term frequency and inverse document frequency and Eq. 3 shows tf-idf weighting calculation. In this study tf-idf weighting is used as term weighting.

$$\text{tf}(\text{d}, \text{f}) = \begin{cases} 0, \text{if}\,(\text{d}, \text{t}) = 0 \\ 1 + \log(1 + \log\,(\text{frequency}\,(\text{d}, \text{t}))) \end{cases} \tag{1}$$

$$\text{idf}(\text{t}) = \log \frac{\text{n}}{|\text{dft}|} \tag{2}$$

$$\text{tf-idf}(\text{d}, \text{t}) = \text{tf}(\text{d}, \text{f}) \times \text{idf}(\text{t}) \tag{3}$$

3.3 Feature Selection and Term Count Threshold

In vector space model contains high dimensional sparse feature vectors. In text mining works every term is represented as feature so this makes vector high dimensional. A RSS news feed is summary of news content so it doesn't have many features but 3337 RSS news feeds generate high dimensional feature space. Classification is hard,

ineffective and time consuming to implement in high dimensional feature space so dimension reduction is necessity. We eliminate stop words and taking only nouns and pronouns therefore we put term count threshold value for training data.

Two common approaches are used in dimension reduction. These are feature selection methods and feature extraction methods. Feature extraction methods combine different features to make new and low dimension feature set. Feature selection methods try to select the best subset of feature set. Two different types of feature selection approach are used in literature. These are wrapper methods and filtering methods. Wrapper methods find best subset with testing all subset combinations. Filtering methods order all features according to filtering approaches. Chi square statistics, document frequency, information gain, mutual information is well known feature selection filtering methods. In similar works [11, 12] chi square statistics and information gain methods give better result than other filtering methods so these two methods are applied separately as feature selection methods in this study.

Chi square statistics and information gain methods are applied to RSS news feeds for dimension reduction. The most successful classification result ,which is equal to 92.2 % F_1-Measure, are taken with using Multinominal Naive Bayes classifier and chi square statistics. 9,477 features are reduced to 1,296 features with chi square statistics method.

Equation 4 shows chi square statistics of t-th feature in class *c*. Chi square statistics value is calculated with occurrence of term in class and absence of the same term in the same class. Chi square statistics give good results in high dimensional sparse feature set.

$$X^2(t, c) = N\left[\frac{\mathrm{P(t, c)} \cdot \mathrm{P(t^{-}c^{-})} - \mathrm{P(t, c^{-})} \cdot \mathrm{P(t^{-}, c)}}{\mathrm{P(t)} \cdot \mathrm{P(t^{-})} \cdot \mathrm{P(c)} \cdot \mathrm{P(c^{-})}}\right]^2 \tag{4}$$

3.4 Retrieval of Test Data

This study aims to evaluate contents of microblogs. 32 bot users' tweets and 30 normal users' tweets are taken as test data. Study also makes comparison between bots and normal users according to their contributions. Categorical information of bots and normal users are taken from wefollow.com. After retrieval of tweets from these two different user types, removal of punctuations and tokenization, which are preprocessing steps, are implemented to test data. Links of images and videos are omitted from tweets. Hashtags are also omitted from tweets. Some tweets consist of only links so after elimination of links make tweets featureless. Featureless tweets or tweets that have one or two features decrease classification success rate. So in test phase description of term count threshold is necessity. We specify three term count threshold values for tweets.

Tweets that are used as test data are arranged according to its term count in testing. Tweets which have more than two terms, three terms and four terms are only evaluated

Table 1 User tweets and term count threshold values

	Term count threshold values		
	>2	>3	>4
Number of normal users' tweets	921	416	162
Number of bots' tweets	2,115	1,039	435

as test data. Table 1 shows that if term counts in tweets and number of tweets which have more than specified term count threshold value is inversely proportional.

This study use only training feature set in training phase and testing phase. After tokenization steps of tweets, features are obtained. If a feature of tweet doesn't occur in RSS news feeds then feature is eliminated from test data set. Tweets have 140 character limitations so microbloggers use abbreviations and nonsense words that belong to social networks. Elimination of words which don't occur in training feature set provides to omit abbreviations and nonsense words so this process enables to make correct classification.

Tweet of normal users and bots are taken in the same time period with RSS news feeds for checking users' up-to-dateness.

3.5 Classification

Multinominal Naive Bayes (MNNB) and Support Vector Machines (SVM) are used as classifier in training and test phases. SVM is popular classifier in text classification area. SVM outperforms k-Nearest Neighbor, Linear Least Square, Naive Bayes, Neural Networks and Decision Methods in terms of classification results [13, 14]. SVM is also good classifier in many other classification areas whose dimensions are high. Multinominal Naive Bayes are especially used in information retrieval and text mining works. It gives good results in these work areas.

3.5.1 Support Vector Machines

Support Vector Machines try to determine the most suitable decision boundary which separate data into their correct classes. The decision boundary must be as far away from data of all class as possible.

Dashed lines show boundaries of each class. Thick line indicates the decision boundary. Samples that are on the dashed lines are called as support vectors. Decision boundaries are determined by support vectors. Data except support vectors has no weights for determination of decision boundaries. Equation 5 shows that for an optimal decision boundary margin (m) must be maximized.

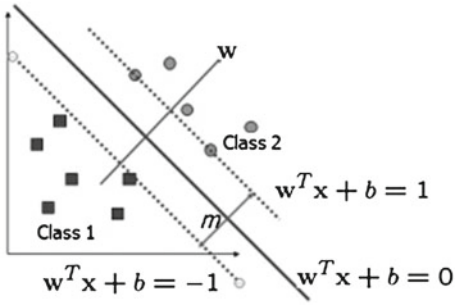


Fig. 2 SVM and decision boundary

$$\mathrm{m} = \frac{2}{||\mathrm{w}||} \tag{5}$$

In other words distance between decision boundary and boundaries of each class is aimed to maximize with minimizing $||\mathrm{w}||$. Assume that class labels of x_i are $y_i \in \{1, -1\}$ and $\{\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3, .., \mathrm{x}_\mathrm{n}\}$ is data set. The decision boundary should classify all data correctly with following constraint that is described in Eq. 6. Equation 6 tries to prevent data from falling into margin.

$$\mathrm{yi}(\mathrm{w}^\mathrm{t}\mathrm{x} + \mathrm{b}) \geq 1 \tag{6}$$

In some cases data can't be separated like Fig. 2 with linear boundary. Data are transformed to higher dimensional space for linear separation with kernel functions. Polynomial, hyperbolic tangent and radial bases functions are used as kernel functions. In this study we prefer linear classifier that generally gives good results.

3.5.2 Multinominal Naive Bayes

Naive Bayes assumes that occurrence of terms are independent from each other. Multinominal Naive Bayes (MNNB) differs from Naive Bayes according to count of term occurrences in text document. Count of term occurrences is used for calculating probability which shows occurrence of term in related class. Equation 7 shows that multiplication of the conditional probabilities for all terms which occurs in the same class gives probability of related class. After probabilities of all classes are calculated, the class which has the highest probability value is selected as correct class among all the probable classes. Equation 8 aims to eliminate zero probability for class so Laplace smoothing is used for it. Class label is given as *c* and term in a text document is given as *t*.

$$\mathrm{P}(\mathrm{c}|\mathrm{d}) = \arg\max_{\mathrm{c} \in \mathrm{C}} \mathrm{P}(\mathrm{c}) \prod_{1 \leq \mathrm{k} \leq \mathrm{V}} \mathrm{P}(\mathrm{t}_\mathrm{k}|\mathrm{c}) \tag{7}$$

$$\mathrm{P}(\mathrm{t}|\mathrm{c}) = \frac{\mathrm{T}_{\mathrm{ct}} + 1}{\sum \mathrm{t}' \in \mathrm{V} \mathrm{T}_{\mathrm{ct}} + \mathrm{B}'} \tag{8}$$

4 Experiment Results and Discussion

Training model is formed by 3337 RSS news feeds. Categorical information of RSS news feeds is given by content suppliers. Four categories are used in this study. These categories are sport, economy, entertainment and technology. RSS news feeds which are preprocessed for text classification are used to form training models by Multinominal Naive Bayes and Support Vector Machines separately.

In testing phase 32 bot users' tweets and 30 normal users' tweets are taken as test data. Categorical information of users is taken from wefollow.com application. Table 2 shows categorical information of user. Number of bot users whose category is sport is higher than other users who have different category. However, numbers of tweets which are taken from bot users whose category is sport are less than other users.

Tweets of two different types of users are given as test data to the training model which is formed by RSS news feeds. Three different term count threshold values are used for test data. These threshold values are more than two, more than three and more than four. More than two means tweets must contain more than two terms, more than three means tweets must contain three terms and more than four means tweets must contain more than four terms. Table 1 shows the number of tweet in terms of threshold values.

F-measure measures for evaluating the performance of classification. F-measure is weighted harmonic mean of precision and recall. Precision and recall weights are taken equal to each other. This is also know F_1 measure. Table 3 gives F-measure values of classification results. First value that is given under the threshold value indicates F-measure of SVM and second indicates F-measure of MNNB. Table 3 shows performance of classification.

Bot users' tweets demonstrate more successful results than normal user's tweets by using tweets that have more than two and three terms. However, F-measure values of bot users' tweets are higher than F-measure value of normal users, classification result of normal users' tweets is higher than classification result of bot users' tweets where SVM and tweets that have more than four terms are used for classification.

Table 3 shows that bot users' tweets are more valuable than normal users' tweets in terms of their categorical information. Contents of bot users reflect their own category more than contents of normal users.

According to the classification performance results, Multinominal Naive Bayes outperforms than Support Vector Machines with any given threshold value and user type. Figure 3 shows the results of Table 3.

Figure 3 and Table 3 shows that using Multinominal Naive Bayes as classifier and tweets of bot users as test data gives the best classification results. Choosing of a classifier affects classification results more than choosing of different types of users' content. Classification performance is also increased by term count threshold value. Selecting of tweets which has more than four terms gives the best classification results with any given classifier. It proves that if a tweet consists of more terms, this makes tweet valuable as test data.

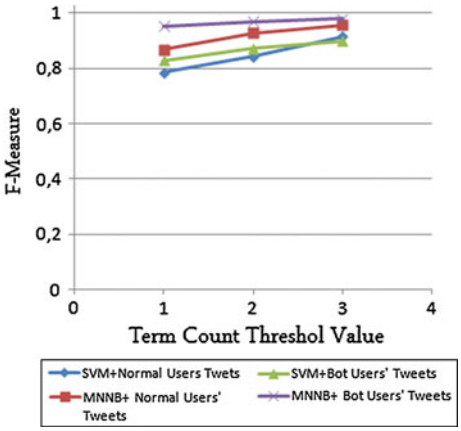
Table 2 Categorical information of users

	Number of normal users	Number of bot users
Sport	8	12
Entertainment	7	7
Technology	7	6
Economy	8	7

Table 3 Classification results, F-measure values (%)

SVM|| MNNB	Term count threshold values					
	>2		>3		>4	
Bot users	82.8	95.2	87.2	96.9	89,9	97.9
Normal users	78.4	86.7	84.2	92.8	91.6	95.7

Fig. 3 Classification results



Rate of correctly classified data is changeable from class to class. Normal users whose categorical information is sport supply more categorical tweets. Rate of correctly classified data is higher than data of other normal users whose categorical information isn't sport. Bot users whose categorical information is economy supply more categorical tweets. Rate of correctly classified data is higher than data of other bots users whose categorical information isn't economy. Both normal users and bot users whose categorical information is technology has the lowest rate of correctly classified data.

5 Conclusion

In our study, we want to check up-to-dateness of users' tweets with using RSS feeds and it is also intended to measure how users reflect their categories.

Bot users' content is more categorical than normal users' content. Classification performance of bot users' tweets are higher than normal users' tweets. 97.9 %

F-measure value can be assumed as good result for microblogs. Microblogs consists of abbreviations and nonsense words because of character limitation so we use only training feature set as complete feature set. In the future, we can collect more than 3337 RSS news feeds for training for precision of classification. After putting term count threshold for tweets it decreases number of tweets which has more terms than threshold values so we can get more tweet for precision of classification.

Working on content mining of microblogs is popular recently. Microblogs reflects microbloggers' thoughts and field of interests. Companies observe content of microbloggers for marketing. Police department also follows contents of microbloggers. Police department observe microbloggers' thoughts and action with using contents of microblogs. Activities of terrorism or crime can be distinguished by this observation. To sum up content mining of microblogs can be used in different areas. Popularity of microblogs is increasing rapidly so works on content mining of microblogs are important for all these different areas.

References

1. Degirmencioglu EA (2010) Exploring area-specific microblogger social networks. Master's thesis, Bogazici University, Turkey
2. Yurtsever E (2010) Sweettweet: a semantic analysis for microblogging environments. Master's thesis, Bogazici University, Turkey
3. Akman DS (2010) Revealing microblogger interests by analyzing contributions. Master's thesis, Bogazici University, Turkey
4. Aslan O (2010) An analysis of news on microblogging systems. Master's thesis, Bogazici University, Turkey
5. Pilavcılar IF (2007) Metin Madenciliğiyle Metin Sınıflandırma. Master's thesis, Yıldız Teknik Universitesi
6. Güç B (2010) Information filtering on micro-blogging services. Master's thesis, Swiss Federal Institute of Technology, Zurich
7. Chua S (2008) The role of parts-of-speech in feature selection. In: The international conference on data mining and applications-IAENG
8. Masuyama T, Nakagawa H (2002) Applying cascaded feature selection to SVM text categorization. In: The DEXA workshops, pp 241–245
9. Lan M, Sung S, Low H (2005) A comparative study on term weighting schemes for text categorization. In: IEEE international conference on neural networks-IJCCN'05
10. Leopold E, Kindermann J (2002) Text categorization with support vector machines. How to represent texts in input space? Mach Learn 46(1–3):423–444
11. Yang Y, Pedersen J (1997) A compartive study on feature selection in text categorization. In: The proceedings of ICML-97
12. Zheng Z, Srihai R (2003) Optimally combining positive and negative features for text categorization. In: ICML workshop
13. Yang Y, Liu X (1999) A re-examination of text categorization methods. In: The proceedings of SIGIR-99, 22nd ACM international conference on research and development in information retrieval Berkeley, US, pp 42–49
14. Joachims T (1998) Text categorization with support vector machiness: learning with many relevant features. In: The European conference on machine learning (ECML)

Glossary

Agent-based Modeling A computational model for simulating the actions and interactions of autonomous individuals or organizations or groups for assessing their effects on the system.

Ant Colony Optimization Algorithm A probabilistic technique, which is fundamentally based on the behavior of ants seeking a path between their colony and a source of food, used for solving computational problems such as community detection.

Brandes' Algorithm A method that determines the shortest paths between all pairs of vertices in a sparse, un-weighted, directed graph and aims to calculate both betweenness and closeness of all vertices in a graph.

Clique Percolation Method A technique for analyzing the overlapping community structure of networks.

Clustering A computational process that places similar items into the same group and dissimilar items into different groups.

Collaborative Filtering A recommender system implementation technique that combines preferences of several users.

Community A group of social actors sharing common properties.

Community Evolution Describes trends of objects and communities whose behavior changes with time.

Community Mining A sub graph, community, identification process in a social network.

Data Mining A computational process that aims to discover hidden patterns in a large data set.

Differential Privacy A privacy model that obscures individuals' contributions to aggregate results in data mining.

Ground Truth A set of data tagged by human experts to prove or disprove research hypotheses.

F. Can et al. (eds.), *State of the Art Applications of Social Network Analysis*, Lecture Notes in Social Networks, DOI: 10.1007/978-3-319-05912-9,

Hadoop HBase A distributed database which is built on top of distributed file system Hadoop.

Machine Learning A branch of artificial intelligence that studies construction of systems that can learn from data.

Micro-blogging Expressing opinions on a blogging service using short text, images or video links.

Naive Bayes A probabilistic machine learning technique used for classification that uses Bayes' theorem with strong/naive independence assumptions.

Network Evolution Similar to community evolution it describes trends of network events such as birth, growth, shrinkage, merge, split, and death.

Recommender System An information filtering system that aims to predict user preferences for several items of interest.

Sentiment Analysis (Opinion Mining) Aims to determine the opinion expressed in a given text with respect to some topic or the general contextual polarity of a document.

Social Actors A set objects of interest in social network analysis, such as humans, organizations, web pages etc.

Social Network A social structure composition with a set social actors and a set of links representing relationships among these actors.

Social Network Analysis Examines the structure of relationships among the actors of a social network using network theory.

Social Network Mining A computation process that involves the use of data mining techniques within the context of social networks.

Support Vector Machines (SVM) A machine learning technique used for classification that aims to separate categories as wide as possible.

Tweet Small messages used in twitter.

Twitter A micro-blogging platform: https://twitter.com/.

Web Forum An online discussion site.