Lecture Notes in Social Networks

Jalal Kawash *Editor*

Online Social Media Analysis and Visualization

Springer

Lecture Notes in Social Networks

More information about this series at http://www.springer.com/series/8768

Jalal Kawash
Editor

Online Social Media Analysis and Visualization

Springer

Editor
Jalal Kawash
Department of Computer Science
University of Calgary
Calgary, AB
Canada

ISSN 2190-5428 ISSN 2190-5436 (electronic)
Lecture Notes in Social Networks
ISBN 978-3-319-13589-2 ISBN 978-3-319-13590-8 (eBook)
DOI 10.1007/978-3-319-13590-8

Library of Congress Control Number: 2014956485

Springer Cham Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media (www.springer.com)

Preface

Online Social Media (OSM) have revolutionized the way people interact and share information. Many recent events and developments have shown that OSM are very powerful tools for people to organize and take action. Examples include the 'Occupy' movement, 'Sandy' relief efforts, and the 'Arab Spring'. OSM have offered a real and viable alternative to conventional mainstream media. The latter is often accused of being biased. Spin and constraints imposed by regulating or funding bodies can hinder mainstream media outlets' unbiased reporting. They often omit (intentionally or otherwise) certain details in their reporting. On the other hand, OSM are likely to provide "raw", unedited information and the details can be overwhelming with the potential of misinformation and disinformation. Yet, OSM are leading to the democratization of knowledge and information. OSM is allowing almost any citizen to become a journalist reporting on specific events of interest. This is resulting in unimaginable amounts of information being shared among huge numbers of OSM participants. As of this writing, twitter claims to have 271 million monthly active users, producing 500 million tweets per day. Facebook grew by the end of 2013 to 1.23 billion users with 757 million users logging on every day. Facebook now has a user base comparable to the population of India! The Daily Mail further reports that the average American daily spends 40 min on Facebook. This is resulting in several billion "likes" and several 100 million posted pictures in a single day.

This book contains 10 contributions that tackle challenges of the subject of OSM analysis and visualization from different angles. These challenges include:

1. Many details are hidden in OSM posts and as a result, search engines tend to favor conventional media sources. Mechanisms that allow for delving deeper into the content of OSM posts so that a more meaningful search can be performed are needed. How can sources of a Twitter event, for example, be accurately identified?
2. Word-of-mouth marketing is a byproduct of OSM. However with OSM, there is the challenge of linking opinions of users to their demographic information. How can we infer such information with the little clues available on a social

network, such as Twitter? Can the gender of users be inferred based on colors used in Twitter profiles?
3. Interests of OSM participants change over time. How can we analyze this change and how can we present it in a convenient manner? What factors are essential for trend-prediction on Twitter, for instance?
4. With the large sizes of users personal networks (an average Facebook user has more than 220 friends), better ways for OSM users are needed in order to better view their social networks. How can the network be visualized in such a way that gives the user immediate important information about friends and posts, such as the level of activity of a friend and the popularity of a post?
5. Making privacy decisions (what to share with whom) is cumbersome given the sizes of personal networks. Going through the list of friends, one-by-one, may no longer be a viable solution. How can we make such privacy decisions aided by a visualization and categorization mechanism offering different privacy levels for different categories?
6. Network analysis is a complex process. This challenge is approached on two fronts: how can we (a) develop realistic models that describe people's interaction and (b) defuse the complexities of the processes through the development of appropriate tool?

Summary of Contributions

Identifying Event-Specific Sources from Social Media

Identifying valuable sources of social media events is a very challenging task, but is a necessary step toward identifying misinformation and disinformation in social media. These sources are often buried in the "long tail." A quick search for some event on major search engines, such as Google, yields top hits for mainstream and conventional media. In addition, conventional media does not often include as much details as social media alternatives. In this chapter, Debanjan Mahata and Nitin Agarwal take on the challenging task of identifying sources from social medial for specific events. The challenges include sparsity of resources, quality assessment, entity extraction, and evaluation measures.

Mahata and Agrawal develop a mutual reinforcement-based methodology for this identification, present an evaluation strategy, validate the approach using real-life data, and conclude with analysis of the model. The empirical evaluation is based on a data set of 11,378 blog posts from different blogging sources. The events are the Egyptian, Libyan, and Tunisian uprisings. Empirical results show that the developed evolutionary mutual reinforcement converges faster with better accuracy than the conventional mutual reinforcement model, and it outperforms Google Blog Search and IceRocket.

Demographic and Psychographic Estimation of Twitter Users Using Social Structures

With the ever expanding number of social media users, such as Twitter and Facebook, many of their posts are geared toward expressing opinions about certain products and services. This can provide a low-cost, real-time word-of-mouth marketing, as opposed to expensive, formal customer surveys. However, formal surveys have the advantage of linking opinions to customer attributes (such as age and gender), but such attributes are often hidden in social media. In this chapter, Jun Ito, Kyosuke Nishida, Takahide Hoshide, Hiroyuki Toda, and Tadasu Uchiyama analyze more than 4.6 million Twitter users in Japan. It is determined that very few users use values in their Twitter profiles that can reveal their age (roughly 3 % described their age), gender (less than 8 % revealed their gender), location (less than 25 % revealed their location), and occupation (less than 14 % indicated their job). Hence, the estimation of these attributes is necessary.

To address this limitation, Ito et al. provide in this chapter a method by which hidden attributes can be estimated from the publicly available information, Twitter profiles and posts, and from social neighbors. Specifically, Ito et al. estimate the four attributes: gender, age, occupation, and interest. The estimation proceeds at three levels. First, a labeling method that identifies users with blog accounts is used to extract their attributes from the blog profile as true labeling for the training data set. Experiments confirm that this is a more accurate labeling than other methods, such as manual labeling and pattern matching. Second, the authors investigate how to combine bag-of-words features of profile documents and tweets. Nine different combining methods are evaluated, identifying the best two of these methods. Finally, information from social neighbors is utilized. Three adjustment levels are investigated.

Say It With Colors: Language-Independent Gender Classification on Twitter

Gender prediction of social network users is important for targeted advertising, law enforcement, and other social reasons. Gender classification in networks such as Twitter heavily depends on analyzing the text of posted messages or tweets. Among the limitations of such an approach is language-dependence, intractability, and non-scalability.

In this chapter, Jalal Alowibdi, Ugo Buy, and Philip Yu present a gender classification approach that is based on colors. This approach is based on analyzing five color features used in Twitter user profiles: background, text, link, sidebar fill, and sidebar boarder colors. Colors are language independent, and using only five features in the analysis (as opposed to millions of features used in text-based classification) makes the color-based approach more desirable for scalability and

tractability. Realizing that the number of colors can be technically enormous, while practically there can be different grades of the same color, Alowibdi et al. empoly a preprocessing step that converts the colors from their RGB (Red, Green, Blue) representation to HSV (Hue, Saturation, Value) representation. The colors are then sorted by their hue and value attributes, providing similar labellings for colors, which are then converted back to their RGB values. This preprocessing step helps improve accuracy and reduce the size of the data set. Empirical results show that the classification accuracy is roughly between 70 and 74 % for different data sets, which is a clear improvement over the 50 % norm. These results are obtained from a data set of about 170,000 users where it was possible to independently verify their genders.

TUCAN: Twitter User Centric ANalyzer

This chapter by Luigi Grimaudo, Han Hee Son, Mario Baldi, Marco Mellia, and Maurizio Munafò takes a text-mining approach to analyzing Twitter posts, using a framework called TUCAN. TUCAN analyzes the tweets of a single, target user over a specific period of time, identifying the interests of that user during the given time window. TUCAN also offers the ability to do a comparison between several users, inferring any common interests. The results are depicted graphically in an intuitive visual representation.

The steps taken in this framework start by projecting a target user's tweets to a time window that Grimaudo et al. call a "bird song". Next, bird songs are filtered and cleaned using known methods in order to eliminate noise and derive general concept terms for the words in the songs. Terms are then scored to identify the important terms for a target user in a bird song. Finally, similarity scores are used to compare two bird songs. The results are provided visually, using colored matrices, where colors distinguish similarity scores.

The authors validate TUCAN by providing an empirical study that includes 740 Twitter users, including 28 public figures, over a period that exceeds two months. This results in analyzing more than 800,000 tweets. Grimaudo et al. also perform a parameter-sensitivity analysis of TUCAN.

Evaluating Important Factors and Effective Models for Twitter Trend Prediction

In this chapter, Peng Zhang, Xufei Wang, and Baoxin answer two important questions related to trend prediction in Twitter. The first question is what content and context factors (or a combination of them) are more important to Twitter trend prediction. The second question is which (if any) is more appropriate for prediction.

To answer these questions, Zhang et al. performed an empirical study using 16.8 million tweets by about 670,000 users over a period of several months. They conducted relevance analysis using tweet content, network topology, and user behavior, addressing the first question. To address the second question, they also performed a prediction performance study for several known prediction models. The analysis concluded that trend factors based on user behavior are more effective in predicting trends, and that the nonlinear state-space models are more suited for prediction.

Rings: A Visualization Mechanism to Enhance the User Awareness on Social Networks

The visualization of someone's social network activities on Facebook is the subject of this chapter by Shi Shi, Thomas Largillier, and Julita Vassileva. The authors take an approach to represent such activities using rings, where the colors and sizes of these rings represent different levels of activities. The result is a tool called *Rings*. Rings allows a user visually and interactively (1) see basic post information, such as the posters' identification and the time; (2) review the activity level of a user; and (3) assess the popularity of posts.

Shi et al. validate Rings using two user studies. The first study assesses if indeed Rings increases user awareness, whether it is useful to users, and how usable the user interface is. The empirical data show general positive results. The second user study delves deeper into Rings validating more specific properties of the system, such as performance, colors, reliability, and other factors.

Friends and Circle—A Design Study for Contact Management in Egocentric Online Social Networks

Due to the large amount of information that a social network user must deal with, managing privacy decisions related to which posts to share with which users becomes an overwhelming process. Users often indicate that they regret certain social network postings. In addition, an average user can easily have more than one hundred friends or followers; for example, a Facebook user had an average of 229 friends in 2013. Going through someone's network, friend-by-friend, in order to decide what level of privacy is required for each friend is not practical. Instead, users tend to categorize their networks allowing a different privacy-level for each category.

Bo Gao and Bettina Berendt target this issue in this chapter by developing an online application, called *FreeBu*, which visualizes a user's network and categorizes his/her friends. Gao and Berendt accomplish this task through several steps.

They present *CircleTree*, a visualization tool that incorporates modularity-based community detection (MOD). A user study is then performed to compare hierarchical MOD with Facebook smart lists. The findings show that hierarchical MOD provides more support for visibility decisions. They also show that graph-based algorithms for community detection are more appropriate than attribute-based algorithms. The authors, then, empirically compare MOD with Generative Model for Friendships (GMF). The study involves ego-networks as follows: 10 from Facebook, 909 from Twitter, and 129 from Google+. Using this data set, it is shown that MOD outperforms GMF. Finally, Gao and Berendt enrich CircleTree with three additional visual interactive methods culminating in FreeBu, exploiting their empirical findings.

Genetically Optimized Realistic Social Network Topology Inspired by Facebook

There is an ever-increasing need to better understand the topological properties of social networks. This requires the development of abstract and generic, and at the same time, flexible and realistic models that describe how people socially interconnect. The synthetic generation of such a topology is very handy to researchers since it provides them with a mechanism to generate social network data with certain specified properties on demand. In this chapter, Alexandru Topirceanu, Mihai Udrescu, and Mircea Vladutiu address this problem of synthetically generating realistic social network topologies. They propose the Genetic-Optimized Social Network (*Genosian*) method. The aim of Genosian is to create accurate replica of friendship models collected from Facebook.

The first empirical observation made by the authors is that realistic Facebook networks share common metrics, in spite of the fact that the networks are diverse in shape and size. It is found that topological metrics of these realistic Facebook networks fall within narrow thresholds. These metrics include: network size, average path length, clustering coefficient, average degree, diameter, density, and modularity. In addition, distribution of degrees, betweenness, closeness, and centrality are looked at.

Genosian uses a Genetic Algorithm (GA) to generate the social network. It starts with a random creation of a collection of communities, inspired by the Watts-Strogatz algorithm. Then, these communities are linked together. The GA optimizes these intra-community edges until the centrality measure of the graph is finessed, aiming at a comparable value to that of real-life Facebook examples. The rewiring of intra-community edges is carried out using GA's natural selection. The empirical results show that on average Genosian produces 63 % more accuracy than the best previous known method.

A Workbench for Visual Design of Executable and Re-usable Network Analysis Workflows

Network analysis is in general a complex process that consists of several steps. Providing social network researchers with tools to assist them in the analysis processes defuses some of these complexities. Tilman Göhnert, Andreas Harrer, Tobias Hecking, and H. Ulrich Hoppe provide in this chapter a social network analysis tool, called *Analytic Workbench*, which offers several advantages over similar tools. The motivation for the Analytic Workbench is rooted in ease of accessibility, support for complex analysis processes in an integrated environment, and explicit representation of analysis workflows. Another motivating factor is the support and ease of integration of additional analysis functions.

The result is a Web-based interface that provides visual representation of multi-step analysis workflows. Explicit representation of analysis workflows yields the ability to reuse workflows, allowing comparative studies with the same analytic methodology. Furthermore, the tool easily provides adaptation of parts of a workflow in another, allowing a researcher to experiment with different algorithms and analytic steps.

Göhnert et al. showcase the Analytic workbench by using blockmodeling analysis on multi-relational networks. The authors also report on a user evaluation study of the tool.

On the Usage of Network Visualization for Multiagent System Verification

In this chapter, Fatemeh H. Fard and Behrouz H. Far take advantage of visualization techniques in order to build an approach for the verification of Multi-Agent Systems (MAS). They make use of social network analysis in order to model the interaction of agents. The purpose of this study is to detect emergent behavior in these networks. An emergent event is an unexpected run-time behavior of the system unforeseen by system designers.

Fard and Far avoid using model checking techniques for detecting emergent behavior due to the scalability drawback of these approaches. Instead with the use of interaction matrices, three networks are derived. The first is a component-level network that describes the agent's behavior. The other two are system-level networks, defining the interaction of agents. The authors illustrate this approach and compare it to other existing approaches through two examples.

Calgary, Canada Jalal Kawash

Contents

Contributors

Nitin Agarwal Department of Information Science, University of Arkansas at Little Rock, Little Rock, USA

Jalal S. Alowibdi Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA; Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

Mario Baldi Narus Inc., Sunnyvale, CA, USA

Bettina Berendt Department of Computer Science, KU Leuven, Heverlee, Belgium

Ugo A. Buy Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA

Behrouz H. Far Department of Electrical and Computer Engineering, University of Calgary, Calgary, AB, Canada

Bo Gao Department of Computer Science, KU Leuven, Heverlee, Belgium

Tilman Göhnert University Duisburg-Essen, Duisburg, Germany

Luigi Grimaudo Politecnico di Torino, Torino, Italy

Andreas Harrer Clausthal Technical University, Clausthal-Zellerfeld, Germany

Tobias Hecking University Duisburg-Essen, Duisburg, Germany

Fatemeh Hendijani Fard Department of Electrical and Computer Engineering, University of Calgary, Calgary, AB, Canada

H. Ulrich Hoppe University Duisburg-Essen, Duisburg, Germany

Takahide Hoshide NTT Service Evolution Laboratories, NTT Corporation, Yokosuka-shi, Kanagawa, Japan

Jun Ito NTT Service Evolution Laboratories, NTT Corporation, Yokosuka-shi, Kanagawa, Japan

Thomas Largillier GREYC Université de Caen - Basse Normandie, Caen, France

Baoxin Li Computer Science and Engineering, Arizona State University, Phoenix, USA

Debanjan Mahata Department of Information Science, University of Arkansas at Little Rock, Little Rock, USA

Marco Mellia Politecnico di Torino, Torino, Italy

Maurizio Munafò Politecnico di Torino, Torino, Italy

Kyosuke Nishida NTT Resonant Inc, Tokyo, Japan

Shi Shi MADMUC Lab, University of Saskatchewan, Saskatoon, Canada

Han Hee Song Narus Inc., Sunnyvale, CA, USA

Hiroyuki Toda NTT Service Evolution Laboratories, NTT Corporation, Yokosuka-shi, Kanagawa, Japan

Alexandru Topirceanu Department of Computers and Information Technology, Politehnica University Timisoara, Timisoara, Romania

Tadasu Uchiyama NTT Service Evolution Laboratories, NTT Corporation, Yokosuka-shi, Kanagawa, Japan

Mihai Udrescu Department of Computers and Information Technology, Politehnica University Timisoara, Timisoara, Romania

Julita Vassileva MADMUC Lab, University of Saskatchewan, Saskatoon, Canada

Mircea Vladutiu Department of Computers and Information Technology, Politehnica University Timisoara, Timisoara, Romania

Xufei Wang Computer Science and Engineering, Arizona State University, Phoenix, USA

Philip S. Yu Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA

Peng Zhang Computer Science and Engineering, Arizona State University, Phoenix, USA

Identifying Event-Specific Sources from Social Media

Debanjan Mahata and Nitin Agarwal

Abstract Social media has become an indispensable resource for coordinating various real-life events by providing a platform to instantly tap into a huge audience. The participatory nature of social media creates an environment highly conducive for people to share information, voice their opinion, and engage in discussions. It is not uncommon to find novel and specific information with intimate details for an event on social media platforms in contrast to the mainstream media. This makes social media a valuable source for event analysis studies. It is, therefore, of utmost importance to identify quality sources from these social media sites for understanding and exploring an event. However, due to the power law distribution of the Internet, social media sources get buried in the Long Tail. The overwhelming number of social media sources makes it even more challenging to identify the valuable sources. We propose an evolutionary mutual reinforcement model for identifying and ranking highly 'specific' social media sources and 'close' entities related to an event. Due to the absence of ground truth, we provide a novel evaluation strategy for validating the model. By considering the top ranked sources according to our model, we observe a substantial information gain (ranging between 25 and 130 %) as compared to the baselines (viz., Google search and Icerocket blog search). Moreover, highly informative sources are ranked much higher according to our model as compared to the widely-used baselines, putting spotlight on the social media sources that could be easily overlooked otherwise. Our model further affords an apparatus to analyze events at micro and macro scales. Data for the research is collected from various blogging platforms such as, Blogger (hosted at blogspot), LiveJournal, WordPress, Typepad, etc. and will be made publicly available for researchers.

Keywords Event analysis · Social media · Blogs · Mutual reinforcement · Specificity · Closeness · Information gain

D. Mahata · N. Agarwal (✉)
Department of Information Science, University of Arkansas at Little Rock, Little Rock, USA
e-mail: nxagarwal@ualr.edu

D. Mahata
e-mail: dxmahata@ualr.edu

J. Kawash (ed.), *Online Social Media Analysis and Visualization*,
Lecture Notes in Social Networks, DOI 10.1007/978-3-319-13590-8_1

1 Introduction

Social media has brought a paradigm shift in the way people share information and communicate. Social media played an important role in mobilizing events such as, 'The Arab Spring', 'Occupy Wall Street', 'Sandy relief efforts', 'London Riots','The Spanish Revolution', among others. This led to a surge in citizen journalism all over the world, encouraging transnational participation. Thus, social media serves as a parallel, yet distinct source of information about real-life events along with the mainstream media space [32].

The mainstream media sources often gloss over the intricate details while covering a real-life event. The information could be biased, regulated by the government, and may not present a well-rounded report of an event [15]. On the contrary, social media sources often contain uninhibited and unedited opinions of the masses. Blogs, especially, are widely accepted in the blogging community as sources of more holistic information with intricate details of an event when compared to mainstream media sources [19]. Thus the sources, which are obtained from social media could potentially provide a rather 'closer' or an on-the-ground view of the events with novel information. The information gleaned from social media affords opportunities to study various social phenomenon from methodological and theoretical perspectives including, situation awareness for better crisis response, humanitarian assistance and disaster relief, social movements, citizen and participatory journalism, collective action [2–4], and more.

Motivation: An initial analysis of the top 10 entities obtained from the top 10 search results related to "Egyptian Revolution" from two mainstream media channels (BBC and CNN), and from blogs during the time of the revolution is shown in Fig. 1. The top entities from the mainstream media channels are certainly relevant but fairly broad level, meaning they do not contribute specific or intricate details about the revolution. In contrast, the top entities from the blogs provide intimate details about the events associated with the revolution. For example, the activists like 'Mona Seif', 'Sarah Carr', 'Maikel Nabil' and 'Hosam El Hamalwy' were very closely involved, and were responsible for mobilizing the event. The entities like 'Internet Blackout' and 'Khaleed Saeed' were central to the event. Moreover, the presence of entities like

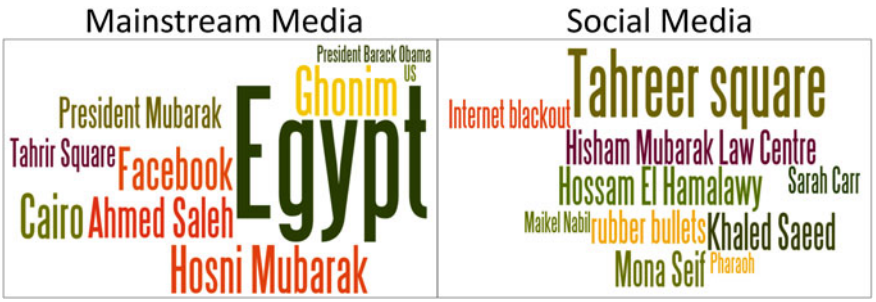


Fig. 1 Top 10 entities from mainstream media and blogs

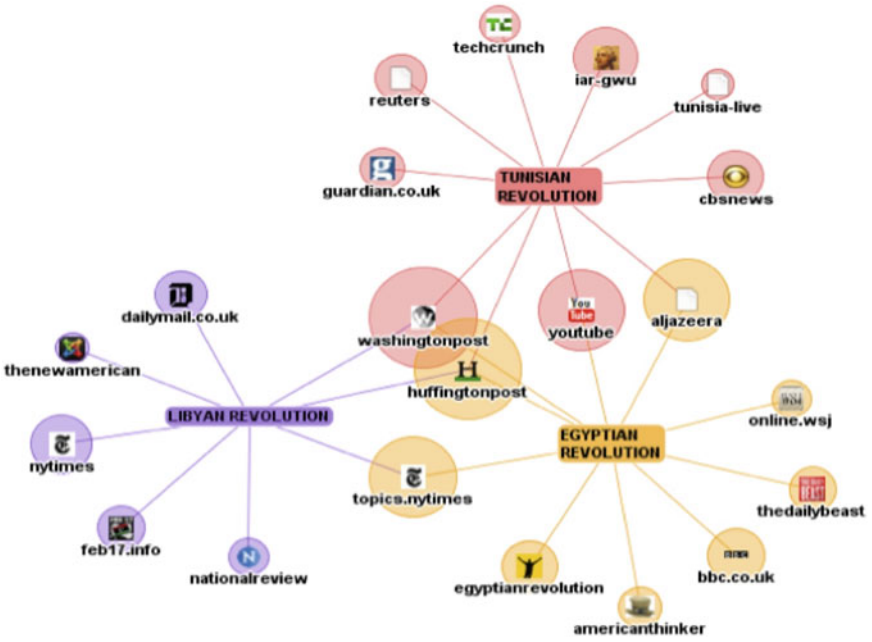


Fig. 2 Top 10 Google search results for "Egyptian Revolution", "Libyan Revolution", and "Tunisian Revolution", visualized using TouchGraph

'Facebook' and 'Ghonim' (who coordinated the event on Facebook) among the top mainstream media entities also indicates the significance of social media in the event.

Due to the power law distribution of the Internet [1], and the present search engine technology, the top search results, or the 'Short Head', is generally dominated by the mainstream media websites. As illustrated in Fig. 2 the top 10 search results for "Egyptian Revolution", "Libyan Revolution", and "Tunisian Revolution" returned by Google, visualized using Touchgraph,[1] retrieved mainstream media sources. Consequently, the social media sites get buried in the "Long Tail" [25] of the search result distribution as shown in Fig. 3. However sources from the social media channels, act as hubs of specific information about real-life events [16]. Thus, a person interested to analyze an event may miss out the novel and specific information available in social media by relying on the top results from the popular search engines. Moreover, in the words of Chris Anderson [6], *With an estimated 15 million bloggers out there, the odds that a few will have something important and insightful to say are good and getting better.* This motivated us to look for techniques in this chapter, that would help in identifying these otherwise buried sources providing highly specific information related to an event.

Challenges: Identifying highly informative 'specific' sources and 'close' entities related to a real-life event from social media entails various challenges as follows,

[1] http://touchgraph.com.

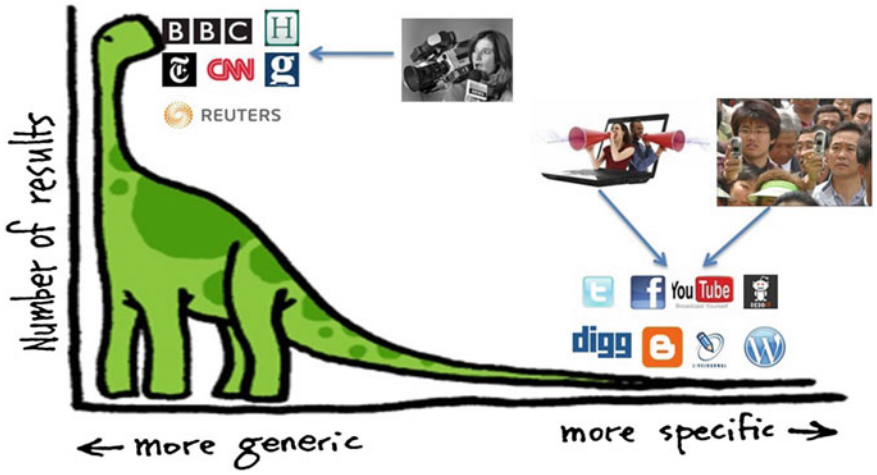


Fig. 3 Short Head versus Long Tail media sources

- **Sparsity of sources**. Enormous population of the sparsely linked Long Tail social media sources
- **Quality assessment dilemma**. The entities (person, organization, place, etc.) mentioned in the sources act as the atomic units of information. Sources which are 'specific' to an event must contain entities 'closer' or highly relevant to the event. On the other hand, such 'close' entities can be obtained from the 'specific' sources. This presents a dilemma in assessing the quality of the sources for event related 'specific' information content, and makes it a nontrivial task.
- **Entity extraction**. It is also a challenge, to accurately extract the entities from the social media sources, which are mostly unstructured and have colloquial content.
- **Lack of evaluation measures**. Conventional information retrieval based evaluation measures help in identifying the most relevant and authoritative sources, however, these sources may not be the most novel or offer specific information. Therefore, new evaluation measures are required to estimate the performance of our work.

Contributions: We make the following contributions,

- **Methodology**. A methodology based on the principle of mutual reinforcement, that helps in identifying highly 'specific' sources and 'close' entities from social media, and their relationships (Sect. 4.2). It ranks the sources and the entities based on their 'specific' information content and how 'close' they are with respect to a set of events.
- **Evaluation Strategy**. Present the methodology, along with an objective evaluation strategy to validate our findings (Sect. 6.3).
- **Experiment on sources related to real-life events**. Perform our experiments on sources and entities related to the events: 'Egyptian Revolution', 'Libyan

Revolution', and 'Tunisian Revolution' (Sect. 6). However, the work is extendible to other types of events.
- **Event analysis**. Explore the utility of such a model in analyzing events (Sect. 7) and conclude the work with future directions (Sect. 8).

Next, we present the related work and compare and contrast these with the proposed approach, highlighting our contributions to the literature.

2 Related Work

Due to huge number of informal sources in social media it is a difficult task to identify high quality sources related to real-life events. Researchers have built semantic web models for efficient retrieval of event related media sources [36]. Event related contents have been found leveraging the tagging and location information associated with the photos shared in Flickr [31]. Becker et al. [7], studied how to identify events and high quality sources related to them from Twitter. In order to identify the genuine sources of information, credibility and trustworthiness of event related information were studied from Twitter [14]. New methods were investigated for filtering and assessing the verity of sources obtained from social media for journalists [10]. All these works, try to explore the quality of information, in terms of relevancy, usefulness, timeliness of the content and usage patterns of authoritative users producing the content. Moreover, none of them involves the blogosphere. However, our work investigates on specific information content in blogs related to a real-life event by using the named entities that are closely associated with the event. The specificity scores of the blogs help in quickly gaining novel and specific information about an event as shown in Sect. 6.3. The method improves the quality of event-specific information gained by users from the ranked sources.

Several methods have been developed in the past for identifying and ranking quality sources from the web [5]. PageRank [9] took advantage of the link structure of the web for ranking web pages. It was further improved for making it sensitive to topic based search [17]. Graph based approaches were used for modeling documents and a set of documents as weighted text graphs, and for computing relative importance of textual units for Natural Language Processing [12]. Mutual reinforcement principle was used for identifying Hubs and Authorities from a subset of web pages using HITS algorithm [21]. The main idea of our algorithm is similar to that of the HITS algorithm. Instead of finding highly authoritative web pages and hubs we find specific sources and entities in the context of an event. Moreover, we propose an evolutionary model that demonstrates faster convergence and better performance as shown in Sect. 6.2. The same principle has been used to solve the problem of identifying reliable users and content from social media [8, 20], as well as tracking discovered topics in web videos [24]. To our knowledge there is no work that explores relationships between named entities and sources from social media for reinforcing the identification of specific information using the Mutual Reinforcement principle.

User-generated data from various social media platforms, related to real-life events, have been studied to perform wide range of analysis. Platforms like TwitterStand [34], Twitris [18], TwitInfo [28] and TweetXplorer [29] have developed techniques to provide analytics, and visualyyizations related to different real-life events. Similar tools have been used for tracking earthquakes [33], providing humanitarian aid during the time of crisis [22], analyzing political campaigns [37] to studying socio-political events [35]. Most of the event analysis frameworks rely on finding relevant keywords and networks between the content producers in order to analyze events. Our work primarily deals with named entities for extracting event-specific information. However, what makes it different from all the other event analysis frameworks is its capability to distinguish highly specific entities from the generic ones among the relevant entities for an event. The entities thus identified helps in further analyzing the event from different perspectives as explained in Sect. 7.

3 Problem Definition

The number of sources related to an event in social media is overwhelming. All these sources may not provide useful information and needs to be processed in order to identify the valuable sources providing specific information about the concerned event. Provided we have a set of events, a set of sources, and a set of entities related to each of these events, we need to rank these sources and entities from the most specific to the most generic ones, based on their information content.

Event: *We define an event to be a real-world incident, occurring at any place at any time or over a certain period of time.*

Specificity and Closeness: *Given a finite set of events* ξ, *we take an event* $E_j \in \xi$ *such that*, $1 \leq j \leq \mid \xi \mid$, *a set of 'p' sources denoted by* ϕ_{E_j}, *and a set of 'q' entities denoted by* σ_{E_j}, *related to the event* E_j. *We define two functions* κ (*specificity*) and τ (*closeness*) such that:

$$\kappa : S_i \rightarrow [0, 1] \tag{1}$$

$$\tau : e_i \rightarrow [0, 1] \tag{2}$$

where, $S_i (\in \phi_{E_j})$, *is the* ith *source, and* $e_i (\in \sigma_{E_j})$ *is the* ith *entity*, *so that we can get two ordered sets* (φ_{E_j} *and* ς_{E_j}) *for the set of sources in* ϕ_{E_j} *and entities in* σ_{E_j}, *such that:*

$$\varphi_{E_j} = \{S_1, \ldots, S_i, S_j, \ldots, S_p \mid \kappa(S_i) \geq \kappa(S_j), i < j\} \tag{3}$$

$$\varsigma_{E_j} = \{e_1, \ldots, e_i, e_j, \ldots, e_q \mid \tau(e_i) \geq \tau(e_j), i < j\} \tag{4}$$

φ_{E_j} is ordered in decreasing order of how 'specific' S_i is w.r.t E_j. ς_{E_j} is ordered in decreasing order of how 'close' e_i is w.r.t E_j. A black-box view of the problem is shown in Fig. 4.

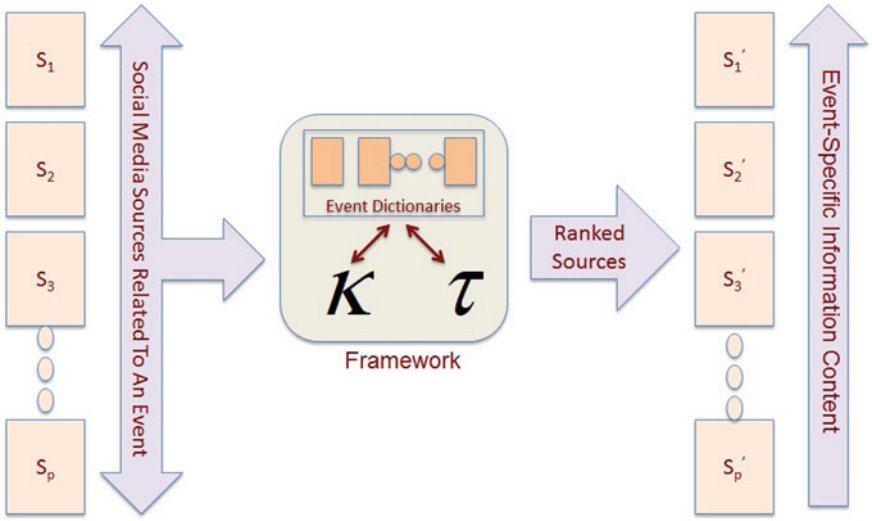


Fig. 4 Black box view of the problem

4 Methodology

A real-life event is characterized by a distinct set of close entities (persons, places, organizations, etc.) along with generic ones. The entities act as the basic units of information in these sources as shown in Fig. 5. Intuitively, specific sources would contain closer entities and one is likely to find closer entities in more specific sources. The relation between specific sources and close entities could then be modeled following the Mutual Reinforcement Principle, which forms the basis of our methodology. The methodology presented here discusses a more rigorous treatment of the problem over our previous studies [26, 27].

> An entity should have high 'closeness' score if it appears in many sources with high 'specificity' scores while a source should have a high 'specificity' score if it contains many entities with high 'closeness' scores.

In essence the principle states that the 'closeness' score of an entity is determined by the 'specificity' scores of the sources it appears in, and the 'specificity' score of a source is determined by the 'closeness' scores of the entities it contain. The proposed methodology extends the basic Mutual Reinforcement Principle to consider the evolving knowledge learned about an event. However, the model requires an apriori or seed knowledge about an event, which is provided in terms of an event profile or an event dictionary. Next, we discuss the construction of event dictionaries.

Fig. 5 Entities associated with a social media source

4.1 Event Dictionaries

Each event E_j is profiled by constructing an event dictionary (σ_{E_j}). In order to calculate specificity of a source w.r.t an event, we need to start with an initial set of close entities. At the same time, these close entities are better acquired from the specific sources. To solve this dilemma, we construct event dictionaries, from independent sources which are completely separate from the sources (ϕ_{E_j}) that need to be ranked. ***Formulation of initial closeness scores***. We calculate the 'closeness' score ($\tau(e_i)_{E_j}$) of each entity (e_i) for event E_j in order to construct the event dictionaries, by using Eqs. 5 and 6 based on tf-idf measure [30] , from the information retrieval literature. Let $E_j \in \xi$, be the jth event, and 'e_i' be the ith entity extracted from the set of sources selected for constructing the event dictionaries. If the term $f(e_i, E_j)$ denotes the frequency of occurrence of the entity 'e_i' in the set of sources for the event E_j, and $IE_jf(e_i)$ denotes the inverse event frequency for the entity 'e_i' then closeness score ($\tau(e_i)_{E_j}$) of an entity e_i w.r.t the event E_j is defined as,

$$\tau(e_i)_{E_j} = e_if\_IE_jf = f(e_i, E_j) * IE_jf(e_i) \tag{5}$$

$$IE_jf(e_i) = \log\left(\frac{|\ \xi\ |}{|\ E_j \in \xi : e_i \in E_j\ |}\right) \tag{6}$$

and, $| E_j \in \xi : e_i \in E_j |$ refers to the number of events in which the entity e_i occurs. Since we extract the entities from the sources related to the events, we cannot have an entity that does not belong to any of the events. Therefore, we always have $| E_j \in \xi : e_i \in E_j | > 0$.

We get $| \xi |$ number of event dictionaries, each corresponding to an event. Following steps are taken to construct the event dictionaries:

1. ***Entity Extraction***. Entities are extracted from all the sources collected from GlobalVoices[2] as explained in Sect. 5, using AlchemyAPI[3] and their corresponding $\tau(e_i)_{E_j}$ values are calculated using Eq. 5. We choose GlobalVoices for obtaining the seed sources for constructing the initial event dictionaries, as it is a portal where bloggers and translators work together to make reports of various real-life events, from blogs and citizen media everywhere.This makes it a reliable source for finding specific information content from social media. Due to colloquial nature of the sources as discussed in the challenges, some of the entities occur in several forms. For example, the entity 'Tahrir Square' occur as 'Tahreer', 'El-Tahrir', etc. We resolve such multiple representation of the same entity by applying pattern matching.[4] Given two entities represented as strings we accept them to be the same if their patterns match by 80 % or more. We would like to use the standard entity resolution algorithms in our future work.
2. ***Closeness Score Computation***. For each event E_j, we calculate $\tau(e_i)_{E_j}$ scores for the set of entities for that event using Eqs. 5 and 6. An entity may occur in multiple events and hence can be present in multiple event dictionaries with different $\tau(e_i)_{E_j}$ scores.
3. ***Ranking***. The higher the $\tau(e_i)_{E_j}$ score of an entity the closer it is to the event. The entities are then ranked according to the descending $\tau(e_i)_{E_j}$ scores.
4. ***Normalization***. Since the range of closeness scores are different for each event, we normalize $\tau(e_i)_{E_j}$ scores w.r.t an event between 0 and 1. The normalization enables an assessment of relative closeness of an entity across multiple events.

The dictionaries thus obtained from the above mentioned procedure are static and serve as a good source of apriori knowledge about the event. However, as we discover new knowledge from specific sources, it is desirable to update the event dictionaries. However, the method applied for constructing the initial event dictionaries require a set of events. This is a drawback of the current method and we plan to improve it in a future work. Next, we discuss how the dictionaries help in identifying specific sources, which in turn help in improving the dictionary.

[2] http://globalvoicesonline.org.

[3] http://alchemyapi.com.

[4] http://docs.python.org/2/library/difflib.html.

4.2 Mutually Reinforcing Sources and Entities

Given an event $E_j \in \xi$, a set of sources (ϕ_{E_j}) and entities (σ_{E_j}), related to the event, we define two column vectors: ***'Specificity'*** ($\kappa_{\mathbf{E_j}}$) and ***'Closeness'*** ($\tau_{\mathbf{E_j}}$).

$$\kappa_{\mathbf{E_j}} = \langle \kappa(S_1)_{E_j}, \kappa(S_2)_{E_j}, \ldots, \kappa(S_p)_{E_j} \rangle^T \tag{7}$$

$$\tau_{\mathbf{E_j}} = \langle \tau(e_1)_{E_j}, \tau(e_2)_{E_j}, \ldots, \tau(e_q)_{E_j} \rangle^T \tag{8}$$

where, $\kappa(S_i)_{E_j}$ ($\in$ *range*(κ), from Eq. 1) represents the *'specificity'* score of ith source $S_i (\in \phi_{E_j})$, for $1 \leq i \leq p$ and $\tau(e_i)_{E_j}$ ($\in$ *range*(τ), from Eq. 2) represents the *'closeness'* score of ith entity $e_i (\in \sigma_{E_j})$, for $1 \leq i \leq q$. Each source S_i may contain related as well as unrelated information about various events. If we consider the set of events ξ, then each $\kappa(S_i)_{E_j}$ is itself a vector of 'specificity' values of the source S_i w.r.t the events ($\in E_j$) as expressed in Eq. 9.

$$\kappa(S_i)_{E_j} = \langle \kappa(S_i)_{E_1}, \kappa(S_i)_{E_2}, \ldots, \kappa(S_i)_{E_{|\xi|}} \rangle \tag{9}$$

Similarly, each entity e_i may be related to various events. If we consider the set of events ξ, then each $\tau(e_i)_{E_j}$ is itself a vector of 'closeness' values of the entity e_i w.r.t the events ($\in E_j$) as expressed in Eq. 10.

$$\tau(e_i)_{E_j} = \langle \tau(e_i)_{E_1}, \tau(e_i)_{E_2}, \ldots, \tau(e_i)_{E_{|\xi|}} \rangle \tag{10}$$

However, while representing $\kappa(S_i)_{E_j}$ and $\tau(e_i)_{E_j}$ as an element of the vectors $\kappa_{\mathbf{E_j}}$ and $\tau_{\mathbf{E_j}}$, respectively, we only choose the entry for the jth event under consideration.

We construct a bipartite graph $G = (V, U)$ (Fig. 6) representing the mutual relationship between the sources and the entities, where $V \in \phi_{E_j}, \sigma_{E_j}$, is the set of vertices for G, and U is the set of undirected edges. The sources without entities are discarded during this process.

The presence of an entity in a source is not sufficient to determine its specificity. In order to express the specificity of a source w.r.t an event, we need to consider the closeness value of the entities present in the source. Given the closeness $\tau(e_n)_{E_j}$ of an entity e_n w.r.t an event E_j, obtained from the event dictionary, a weight w_{mn} is assigned to the edges of the graph, which expresses the magnitude by which an entity is related to a source S_m.

The significance of an entity e_n in a source S_m is expressed as,

$$\frac{f(e_n, S_m)}{\sum_{n=0}^{q} f(e_n, S_m)} \tag{11}$$

where, $f(e_n, S_m)$ is the frequency of occurrence of the entity e_n in the source S_m. Therefore mathematically,

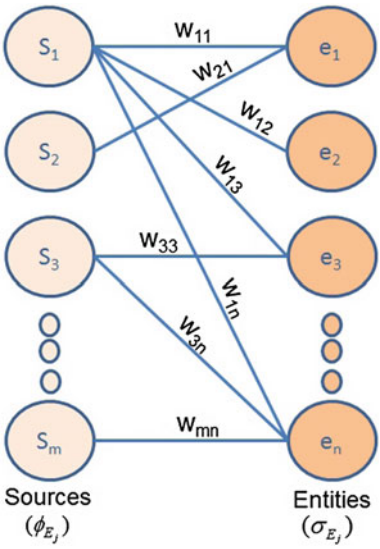


Fig. 6 Bipartite graph G representing the mutual relationship between the sources and the entities

$$w_{mn} = \frac{\tau(e_n)_{E_j} * f(e_n, S_m)}{\sum_{n=0}^{q} f(e_n, S_m)} \tag{12}$$

The adjacency matrix of the bipartite graph G is denoted by L, and is defined as follows:

$$L_{mn} = \begin{cases} w_{mn} & if\ (m,n) \in U \\ 0 & otherwise \end{cases}$$

Following the Mutual Reinforcement Principle the relationships between specificity scores of sources and closeness scores of entities for event E_j can be denoted as follows,

$$\kappa_{\mathbf{E_j}} = \mathbf{L}\tau_{\mathbf{E_j}} \tag{13}$$

$$\tau_{\mathbf{E_j}} = \mathbf{L^T}\kappa_{\mathbf{E_j}} \tag{14}$$

Substituting the values for $\kappa_{\mathbf{E_j}}$ and $\tau_{\mathbf{E_j}}$, we derive the following equations,

$$\kappa_{\mathbf{E_j}} = \mathbf{L}\mathbf{L^T}\kappa_{\mathbf{E_j}} \tag{15}$$

$$\tau_{\mathbf{E_j}} = \mathbf{L^T}\mathbf{L}\tau_{\mathbf{E_j}} \tag{16}$$

Equations 15 and 16 are characteristic equations of an eigensystem, where the solutions to $\kappa_{\mathbf{E_j}}$ and $\tau_{\mathbf{E_j}}$ are the respective eigen vectors with the corresponding eigenvalue of 1.

To emphasize the relationship between the sources and the entities, we make a major contribution by modifying the way the Eqs. 15 and 16 are solved. We make the matrices $\mathbf{LL^T}$ and $\mathbf{L^TL}$ evolutionary while solving the equations. Since each of the equations is a circular definition, the final specificity and closeness scores are computed using the power iteration method [13]. Each iteration improves specificity and closeness scores reflecting their mutual relationship. As we move towards getting the specific sources and close entities in each iteration, we update the weights (w_{mn}) assigned to the edges between the sources and the entities by the newly calculated closeness scores for the entities. This results in renewed reinforcement of the relationship at every iteration by getting closer entities from better sources and vice-versa. This essentially helps the model incorporate the newly discovered knowledge about the events. More precisely, the improved understanding of the relationship between the source and the entities vis-a-vis an event is incorporated into the model.

The updation of the edge weights and the matrices with kth iteration is represented as follows,

$$w_{mn(k)} = \frac{\tau(e_{n(\mathbf{k-1})})_{E_j} * f(e_n, S_m)}{\sum_{n=0}^{q} f(e_n, S_m)} \tag{17}$$

$$L_{mn(k)} = \begin{cases} w_{mn(k)} & if\ (m,n) \in U \\ 0 & otherwise \end{cases}$$

where, $L_{mn(k)}$ represents the adjacency matrix for graph G, and $w_{mn(k)}$ denotes the edge weight for the edge between mth source and nth entity at the kth iteration. $\tau(e_{n(\mathbf{k-1})})_{E_j}$ represents the closeness score of the entity e_n w.r.t the event $E_j(\in \xi)$, obtained from the evolving event dictionary for event E_j at $(k-1)$th iteration.

If, $\kappa_{\mathbf{E_j}}(\mathbf{k})$ and $\tau_{\mathbf{E_j}}(\mathbf{k})$ be the specificity and the closeness scores, at the kth iteration, the iterative process for generating the final solution are,

$$\kappa_{\mathbf{E_j}}(\mathbf{k}) = \mathbf{L_{k-1}L_{k-1}^T}\kappa_{\mathbf{E_j}}(\mathbf{k-1}) \tag{18}$$

$$\tau_{\mathbf{E_j}}(\mathbf{k}) = \mathbf{L_{k-1}^TL_{k-1}}\tau_{\mathbf{E_j}}(\mathbf{k-1}) \tag{19}$$

In order to get 1 as the largest eigenvalue and, $\kappa_{\mathbf{E_j}}$ and $\tau_{\mathbf{E_j}}$ as the principal eigen vectors, the matrices $\mathbf{L_{k-1}L_{k-1}^T}$ and $\mathbf{L_{k-1}^TL_{k-1}}$ needs to be stochastic and irreducible [23] at every step of our evolutionary process. In the present case, since the graph G is a bipartite graph, matrices $\mathbf{L_{k-1}L_{k-1}^T}$ and $\mathbf{L_{k-1}^TL_{k-1}}$ are already irreducible.

In order to make the matrices $\mathbf{L_{k-1}L_{k-1}^T}$ and $\mathbf{L_{k-1}^TL_{k-1}}$ stochastic, we take the following steps at each iteration,

- Dividing the non-zero entries of the matrices $\mathbf{L_{k-1}L_{k-1}^T}$ and $\mathbf{L_{k-1}^TL_{k-1}}$ by the summation of all the entries in a row.
- Assigning $1/n$ to the zero entries of $\mathbf{L_{k-1}L_{k-1}^T}$ and $1/m$ to the zero entries of $\mathbf{L_{k-1}^TL_{k-1}}$, respectively.

The whole process is presented as an algorithm as shown in Fig. 7.

Input: Set of sources ϕ_{E_j}, set of entities σ_{E_j} from the event dictionary, threshold for convergence μ.
Output: Ordered set of sources φ_{E_j}, ranked according to their 'specificity' scores w.r.t E_j, and, ordered set of entities ς_{E_j}, ranked according to their 'closeness' scores w.r.t E_j.

Initialize $\kappa_{\mathbf{E_{J(0)}}} \leftarrow \tau_{\mathbf{E_{J(0)}}} \leftarrow < 1, 1, ..., 1 >$;
Initialize $k \leftarrow 1$;
repeat
  Construct matrices $\mathbf{L_{k-1}}$ and $\mathbf{L^T_{k-1}}$;
  Calculate matrix products $M \leftarrow \mathbf{L_{k-1}L^T_{k-1}}$ and $M'' \leftarrow \mathbf{L^T_{k-1}L_{k-1}}$;
  Convert M and M'' into stochastic matrices
  $M_{stochastic} \leftarrow M$ *and* $M''_{stochastic} \leftarrow M''$;
  $\kappa_{\mathbf{E_J}}(\mathrm{k}) \leftarrow M_{stochastic}\kappa_{\mathbf{E_J}}(\mathrm{k}-1)$;
  $\tau_{\mathbf{E_J}}(\mathrm{k}) \leftarrow M''_{stochastic}\tau_{\mathbf{E_J}}(\mathrm{k}-1)$;
  normalize $\kappa_{\mathbf{E_J}}(\mathrm{k}) \leftarrow \frac{\kappa_{\mathbf{E_J}}(\mathrm{k})}{||\kappa_{\mathbf{E_J}}(\mathrm{k})||_1}$;
  normalize $\tau_{\mathbf{E_J}}(\mathrm{k}) \leftarrow \frac{\tau_{\mathbf{E_J}}(\mathrm{k})}{||\tau_{\mathbf{E_J}}(\mathrm{k})||_1}$;
  $k \leftarrow k + 1$;
  Update σ_{E_j} ;
until $|| \kappa_{\mathbf{E_J}}(\mathrm{k}) - \kappa_{\mathbf{E_J}}(\mathrm{k}-1) ||_1 < \mu$ *and* $|| \tau_{\mathbf{E_J}}(\mathrm{k}) - \tau_{\mathbf{E_J}}(\mathrm{k}-1) ||_1 < \mu$;
Reverse sort $\kappa_{\mathbf{E_J}}(\mathrm{k})$,$\tau_{\mathbf{E_J}}(\mathrm{k})$;
return $\kappa_{\mathbf{E_J}}(\mathrm{k})$,$\tau_{\mathbf{E_J}}(\mathrm{k})$;

Fig. 7 Algorithm for calculating 'specificity' and 'closeness'

We also perform our study using conventional binary static matrices represented as follows,

$$L_{mn} = \begin{cases} 1 & if\ (m,n) \in U \\ 0 & otherwise \end{cases}$$

The proposed evolutionary model outperforms the static model as validated by the results discussed in Sect. 6.

5 Data Collection

Motivation behind source selection. For many people blogs have become popular social media sources for satisfying interpersonal communication needs. Blogs act as a platform for masses to share their likes and dislikes, voice their opinions, provide suggestions and report news. Over the years blogging has matured from personal diaries to citizen journalistic sources providing live coverage of events beyond the

professional newsrooms. Often mainstream media rely on blogs for reporting first-hand accounts of an event [11]. Other social media platforms like microblogs, social networks etc., also promote such activities. But, these platforms have very little scope to elaborately discuss about the events due to the limitations in the length of content allowed to be posted. However, these alternative platforms act as good sources for studying and tracking dissemination of information during real-life events. This motivated us to perform our experiments on sources collected from the blogging platforms instead of other social media websites.

Sources. Blog posts from GlobalVoices, Blogger[5] and Icerocket Blog Search[6] respectively, are collected for the study. The details of the dataset used is given in Table 1. The dataset includes 11,378 blog posts from various blogging platforms like blogspot.com, wordpress.com, livejournal.com, typepad.com, etc. We also filter out the non-English blogs. The data from GlobalVoices is used for constructing event dictionaries (σ_{E_j}), as explained in Sect. 4. We collect blog posts related to the three events from Blogger using Google search, and from other blogging platforms using Icerocket blog search. We perform our experiments on the sources (ϕ_{E_j}) retrieved by the search engines due to the lack of ground truth and take the sources along with the ranks assigned to them by the search engines as our baseline (explained in Sect. 6.3) The collected blog posts are parsed for extracting various information. However, we use the following information for our study: *URL* of the blog and blogpost), *blog text*, *entities*, *language*, and *rank* of the post in the respective search engines used for collecting it. We use AlchemyAPI in order to extract entities. These datasets would be made available on request.

Table 1 Details of data collected

Service Used	Event	Number of blog posts
GlobalVoices	Egyptian Revolution	234
	Libyan Revolution	86
	Tunisian Revolution	77
Google Blogger	Egyptian Revolution	579
	Libyan Revolution	600
	Tunisian Revolution	484
Icerocket Blog Search	Egyptian Revolution	5,900
	Libyan Revolution	2,198
	Tunisian Revolution	1,220

[5] http://blogger.com.

[6] http://icerocket.com.

6 Experiment and Analysis

In this section, we describe the experiments performed. First, we discuss the experimental setup, followed by the comparative analysis between the proposed evolutionary mutual reinforcement model and conventional mutual reinforcement model. We then, introduce a novel evaluation strategy comparing the proposed model with two baseline models.

6.1 Experimental Setup

The methodology discussed earlier is implemented on the collected datasets. We take the following steps in order to perform the experiment,

- ***Constructing the Event Dictionaries***. We take ξ = {"Egyptian Revolution", "Libyan Revolution", "Tunisian Revolution"} as our set of events. We construct the event dictionaries (σ_{E_j}) by using the sources from GlobalVoices (explained in Event Dictionary subsection).
- ***Implementing the Proposed Evolutionary Mutual Reinforcement Model***. The algorithm, as presented in Fig. 7, is implemented on the set of sources (ϕ_{E_j}) from Blogger and Icerocket related to each event respectively, and the set of entities (σ_{E_j}) from the event dictionary corresponding to each event E_j. The threshold value for convergence μ is set to 1e-08.
- ***Obtaining Specific Sources and Close Entities***. With the termination of the algorithm we get a ranked set of sources (φ_{E_j}) ordered in terms of their specific information content and entities (ς_{E_j}) ordered in terms of how closely related they are to the event.
- ***Conventional Mutual Reinforcement Model Approach***. We also implement the conventional mutual reinforcement model on ϕ_{E_j} and σ_{E_j} without considering the evolving matrices (explained in Sect. 4.2).

6.2 Comparing Conventional and Evolutionary Mutual Reinforcement Models

In order to compare the efficiency of the proposed evolutionary mutual reinforcement model with the conventional mutual reinforcement model we combine the sources collected for an event E_j from Google search and Icerocket search. After that we run the proposed evolutionary mutual reinforcement model and conventional mutual reinforcement model on the combined set of sources and event dictionary (σ_{E_j}) for each event. The number of iterations taken by the power iteration method to converge in each case is analyzed in Fig. 8. It is observed that the number of iterations taken by the power iteration method is lesser, when we employ the evolutionary mutual

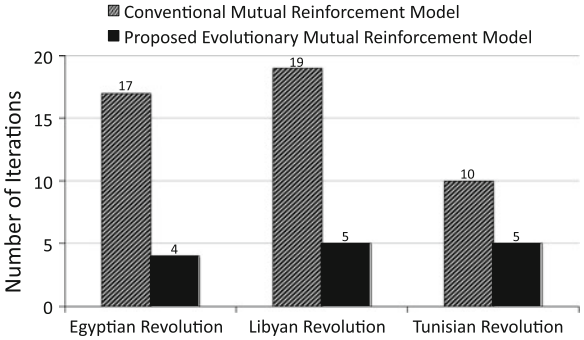


Fig. 8 Comparison of number of iterations taken by the power iteration method to converge, for the set of sources related to events in ξ, with the proposed evolutionary mutual reinforcement model and the conventional static mutual reinforcement model

reinforcement model. Hence, we observe a marked improvement in the performance of our evolutionary mutual reinforcement model over the conventional static mutual reinforcement model.

Explanation for the improvement. The lesser number of iterations in the proposed evolutionary mutual reinforcement model can be explained by the introduction of the evolutionary weights assigned to the relationship between the sources and the entities. The improved closeness scores ($\tau(e_i, E_j)$) at each iteration of the evolutionary model and the renewed reinforcement of the relationship between the entities and the sources decreases the number of iterations taken by the proposed evolutionary model to converge in comparison to the static conventional model. Next, we compare the performance of the proposed evolutionary model with the baselines and the conventional mutual reinforcement model in terms of how quickly the models help in identifying valuable information about the events.

6.3 Baseline Comparisons

Selecting the Baselines. Due to lack of benchmark datasets we use the search results obtained from Google and Icerocket Blog Search as baselines for validation. Standard information retrieval measures for evaluation (DCG, NDCG, MAP, MAP@10) could not be used due to the absence of ground truth. We also consider the sources ranked according to the conventional static mutual reinforcement model and show the effectiveness of our model in quickly gaining information about an event. We further analyze the ranking of the top K specific sources as identified by our methodology and observe the difference in their rankings as assigned by the search engines and as assigned by our model. Figure 9 shows ranks assigned by Google and Icerocket for the top 10 sources for each event, ranked according to our framework. We conclude, that our framework could identify the sources that often gets buried

Egyptian Revolution

Specificity Based Ranking	Google Search Ranking
1	59
2	286
3	400
4	277
5	55
6	202
7	6
8	9
9	313
10	374

Libyan Revolution

Specificity Based Ranking	Google Search Ranking
1	13
2	329
3	9
4	194
5	24
6	311
7	364
8	204
9	374
10	184

Tunisian Revolution

Specificity Based Ranking	Google Search Ranking
1	162
2	40
3	420
4	459
5	72
6	181
7	152
8	440
9	99
10	174

Egyptian Revolution

Specificity Based Ranking	Icerocket Blog Search Ranking
1	75216
2	10607
3	53924
4	56604
5	9831
6	25790
7	1
8	99925
9	94614
10	53924

Libyan Revolution

Specificity Based Ranking	Icerocket Blog Search Ranking
1	47276
2	11751
3	4900
4	22501
5	4
6	11040
7	43520
8	11751
9	41631
10	18271

Tunisian Revolution

Specificity Based Ranking	Icerocket Blog Search Ranking
1	9713
2	42985
3	36335
4	3843
5	46784
6	42645
7	99
8	1
9	63141
10	42645

Fig. 9 Rankings of the sources from Google Blogger and Icerocket based on 'specificity' (κ) values obtained from our model and the rankings assigned by Google Search and Icerocket Blog Search

in the Long Tail and has the potential for presenting valuable information about the event. Next, we propose a novel strategy to demonstrate the effectiveness of the specific sources ranked by our evolutionary mutual reinforcement model in identifying highly informative sources.

Rationale Behind The Strategy. Search engines are designed to give the most relevant sources containing the close entities, related to a query for a given event. As these close entities have a very high probability to be associated with the event, we expect to gain valuable information about the event from these entities making the highly ranked sources very specific to the event. We use this notion to propose a novel evaluation strategy, showing that when sources related to an event are ranked according to our model, they provide more valuable information than the ranking order given by the search engines and the conventional mutual reinforcement model for the same set of sources.

Implementation of The Strategy. We compare closeness ($\tau(e_i)_{E_j}$) values of the entities obtained from the sources ranked according to the search engines, the conventional model and our model respectively. Following steps are taken,

- ***Preparing the Ranked Lists of Entities***. From the three differently ranked lists (search engine, our evolutionary model and conventional static model), each source is visited and the entities are extracted from them. The '$\tau(e_i)_{E_j}$' values are assigned to these entities by referring to the respective final event dictionary (ς_{E_j}) and they are ranked in descending order.
- ***Measuring the Information Gain***. As we traverse the three list of sources, we obtain three list of same entities, arranged in different orders depending upon the ranking of the sources. In order to show the comparison in the gain of information from the three lists we take the top 'K' entities from each list and calculate the sum of their '$\tau(e_i)_{E_j}$' values and plot them against the value of 'K' in Fig. 10. We start from $K = 1$ and go on increasing its value till the number of entities are exhausted in all the three lists. It is evident from Fig. 10, that the curves based on specificity ($\kappa(S_i)_{E_j}$) quickly gains over the curves based on the search engines and the conventional model.
- ***Analysis***. Figure 10 shows that information about the event is gained quicker using our model, which could identify specific sources earlier than the search engines as well as the conventional model. We measure the maximum percentage gain of information in each set of sources related to the three events. There is a maximum gain of information (130.4 %) in case of sources obtained from Icerocket search related to 'Tunisian Revolution', and a minimum gain of information (25.97 %) in case of the sources obtained from Icerocket search related to 'Egyptian Revolution'. Also when the sources are ranked according to our methodology they gain the maximum percentage of information at the 9th source in case of sources obtained from Google related to 'Libyan Revolution'. This in turn implies that the sources ranked higher by our model are more specific than the ones ranked by the search engines and the conventional model. These highly specific sources are also very informative. When presented earlier they also help in learning useful information about the event due to the presence of close entities in them. As we already observed earlier that these sources are often Long Tail sources, we can conclude that Long Tail sources that are ranked lower by the search engines, when identified are often more specific than the highly ranked short head sources.

7 Further Exploration

We use the proposed framework developed by us as an apparatus to further explore event characteristics and show its utility in analyzing events. We show the potential of the final event dictionaries (ς_{E_j}) obtained for each event E_j for gaining valuable information about the event and to identify the generic entities related to the class of events.

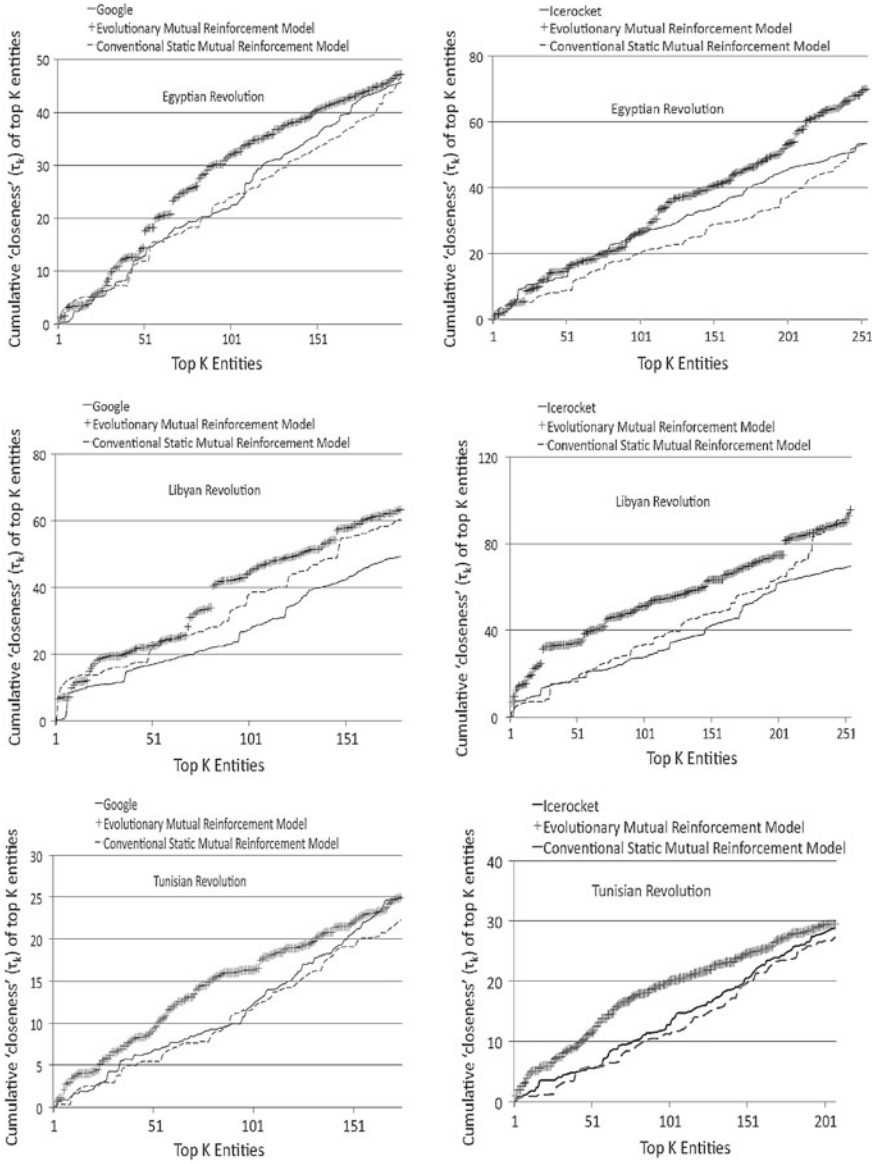


Fig. 10 Validating specific sources obtained from our model

The entities in the final event dictionaries (ς_{E_j}) are examined further. Based on their frequency $f(e_i, E_j)$ and closeness ($\tau(e_i)_{E_j}$) scores, we categorize the entities into the following categories: a. *Close and Frequent*, b. *Close but Not Frequent*,

c. *Not Close but Frequent*, and d. *Neither Close nor Frequent* as shown in Fig. 7. We categorize an entity in each event dictionary as 'frequent' if it occurs more than a threshold value η i.e., $f(e_i, E_j) \geq \eta$, and 'close' if it has a closeness score more than a threshold value α i.e., $\alpha \geq \tau(e_i, E_j)$. After careful manual inspection we decided the values of η and α to be 5 and 0.5 respectively. The values of η and α is same for all the events $E_j \in \xi$. However, different thresholds could be examined in the future though. Each entry in a matrix, consists of top 10 entities that satisfy the thresholds for the corresponding row and column category. We further examine these entities in the context of each of these events and report interesting observations. We present the observations for "Egyptian Revolution", next, however similar observations were made for other events.

7.1 Event-Specific Popular and Close Entities

We analyze the entities in the *Close and Frequent*, and *Close but Not Frequent* categories. We find that *Close and Frequent* entities are not only frequent but are also closely related to the event. In other words these are very popular entities related to the event. For example, the occurrence of 'Tahrir Square' and 'Egyptian government' in this category, is inevitable, as Tahrir Square is the place where the protest started against the Egyptian Government. These are also the entities that anyone comes to know about from the top search results given by the search engines as well as from mainstream media sources.

On the other hand, *Close but Not Frequent* category primarily consists of entities that add novel and useful insights to the events. The occurrence of 'Internet Cafe' in this category clearly shows how the local people in Egypt used Internet Cafes for accessing various social media websites in order to coordinate and participate in the revolution. It is also the place where 'Khaled Said', a 28 year old computer-wiz was arrested by the Egyptian police. He was brutally tortured to death, triggering the anger among the people of Egypt for a mass uprising against the dictatorial government. January 25, 2011 is also known as 'The Day of Anger' in the Egyptian Revolution, as it is the day that marked the start of a series of protests and riots in Egypt. So the occurrence of the entity, 'Day of Anger' in this category is reasonable.

Drawing upon the differences between the two categories our findings suggests that although the category of *Close and Frequent* entities give a lot of information about an event, it is also necessary to know about the entities of *Close but Not Frequent* category. The entities belonging to *Close but Not Frequent* category provides in-depth information about the event from the grass root level. The *Close but Not Frequent* entities are likely to be found buried in the Long Tail sources. In order to gain maximum information about an event it is necessary to identify the entities from both the categories. They point to the key persons, places and organizations related

to the event. In other words, these entities when present in a source makes it highly specific to the event. Both these categories of entities can prove to be vital sources of information about the event.

7.2 Event-Specific and Event-Class Specific Dictionaries

Based on the categorization and the analysis conducted in the previous subsection we divide the final event dictionaries (ς_{E_j}) for each event into *event-specific* and *event-class specific* dictionaries. The *event-specific* dictionaries are comprised of the entities categorized as *Close and Frequent*, and *Close but Not Frequent* for each event $E_j \in \xi$ respectively. Whereas, the *event-class specific* dictionary is comprised of the entities found in the *Not Close but Frequent* and *Neither Close nor Frequent* categories for all the events $E_j \in \xi$ (Fig. 11).

Table 2 shows the top five entities in the *event specific* dictionaries for each event E_j and a socio-political event dictionary for the set of events ξ under study. The entities in the *event specific* dictionaries, when present in a source makes it highly specific to that particular event and contributes in gaining information about it. These entities can also help in conducting micro-analysis of an event by identifying precise details about the event. On the other hand, the entities of the *event-class specific* dictionary provides shallow information about a specific event, and are useful in learning about a category of events, in this case, socio-political uprisings in the middle east. These entities can help in conducting macro-analysis of an event by classifying the event into a certain category (like socio-political, crisis, entertainment, economic, etc.) and point out the general characteristics of the event. However, the analysis requires a set of known events, which could be a perceiveable constraint. We plan to address this in the future work.

Table 2 Top 5 entities in the event specific and the event class dictionaries constructed for the set of events ξ

Egyptian Revolution Specific Dictionary	Tahrir Square, Egyptian government, Gigi Ibrahim, Alexandria, Wael Abbas
Libyan Revolution Specific Dictionary	Tripoli, Muammar Al Gaddafi, North Atlantic Treaty Organization, Chad, United Kingdom
Tunisian Revolution Specific Dictionary	Tunisian government, Lin Ben Mhenni, Samir Feriani, Kasbah Square, RCD
Socio-Political Event Dictionary	Twitter, Iranian Government, Tear gas devices, Facebook, Big Social network

	Frequent (η >= 5)	(η < 5) Not Frequent
(α >= 0.5) Close	Egyptian government, Tahrir Square, Suez, Gigi Ibrahim, Alexandria, Maikel Nabil, NDP, Muslim Brotherhood, Egyptian Army, Wael Ghonim.	Internet Café, Day of Anger, Mariam Arafat, Candice Holdsworth, Dalia Al Marghani, Sherine Tadros, EGP, Bahaa El-Tawil, Mir Hussein Mousavi, Hussein Sharif.
Not Close (α < 0.5)	Anwar al-Sadat, Open Society Institute, Facebook, Professor Rashid Khalidi, dictator, Mohammed Abdel Dayem, mass movement, illegal occupation, Tea Party, Abdel Salam Karmen.	Mainstream media, Oman, Tunis, Carlos Latuff, Ukraine, Sudan, Cuba, Hillary Clinton.

Egyptian Revolution

	Frequent (η >= 5)	(η < 5) Not Frequent
(α >= 0.5) Close	Saif Al Islam Gaddaffi, Pentagon, oil, Mohamed Nabbous, Central Africa, Muammar Gaddafi, Rwanda, Arab League, Ethiopia, Khamis Gaddaffi	North Atlantic Treaty Organization, Iyad El Baghdadi, Libyan State Television, Bent Benghazi, Altarash, Benina airport, Omar Al-Mukhtar, Youssef Al Qaradawi, Eman Al Obaidi, Hamdi Kadri
Not Close (α < 0.5)	Tony Buckingham, Senoussis Brotherhood, Safia Farkash, Cynthia McKinney, United States of America, Canada, popular media, United Africa, Emad Benosman, Hosni Mubarak	Mainstream media, Oman, Tunis, Bahrain, UAE, Saudi Arabia, New York Times, Afghanistan, Ben Ali, Wael Ghonim

Libyan Revolution

	Frequent (η >= 5)	(η < 5) Not Frequent
(α >= 0.5) Close	Ennahda party, Moncef Marzouki, Sidi Bouzid, Tunisian government, Habib Bourguiba, Al Abedine Ben Ali, Muslim Brotherhood, RCD, Mohammed Ghannouchi.	Kasbah Square, France, Slim Amamou, Amira Yahyaoui, Beji Caid Al Sebsi, Mehdi Lamloum, Sami Ben Gharbia, Aziza Othmana Hospital, Samar Dahmash Jarrah, Sami Ben Romdhane
Not Close (α < 0.5)	Michele Alliot-Marie, social networks, facebook, Arabs, Jeffrey Feltman, United States of America, Al-Mahdi, Al-Qaida, Al Jazeera, Kareem Salama	Mainstream media, Oman, Doha, Tunis, UAE, Saudi Arabia, New York Times, Hosni Mubarak, Bill Clinton, Hillary Clinton

Tunisian Revolution

Fig. 11 Different categories of entities for the events

8 Conclusions and Future Work

In this chapter, we highlighted the need for exploring the social media sources to study an event. We demonstrated that social media sources that are often buried in the Long Tail have the capability to provide very specific and novel information. However, the sheer volume of social media sources and the Long Tail characteristics (e.g., link sparsity, colloquial language, etc.) make it extremely challenging to identify the specific sources. Towards this direction, we developed a methodology that utilizes relevant entities as a mechanism to identify specific sources using a mutual reinforcement framework. Further, in order to consider the dynamic relationship between the specific sources and close entities, an evolutionary mutual reinforcement model is developed. Experiments conducted on real-world datasets for the three social movements during the Arab Spring, viz., Egyptian Revolution, Libyan Revolution, and Tunisian Revolution demonstrate faster convergence and better accuracy of the evolutionary mutual reinforcement model over the conventional mutual reinforcement model. Furthermore, the evolutionary mutual reinforcement model outperformed one of the most-widely used search engines, i.e., Google Blog Search and IceRocket. It was observed that the search engines ranked the specific sources surprisingly low, thereby reducing the chances of their discovery. The poor hyperlink connectivity of these Long Tail social media sources was contemplated to be a primary reason behind their low ranks in traditional search engines. By analyzing the close entities identified by our model we also showed the potential of the framework to be utilized for analyzing events. As next steps, we plan to systematically study the applicability of the model in other social media platforms, especially the microblogging sites (e.g., Twitter).

Recently, social media sources are increasingly found to be plagued with false information that poses challenges for information and knowledge extraction for an event. The instances of false information are categorized as misinformation (i.e., inadvertent dissemination of false information due to relying upon inadequately validated facts, evolving nature of details during breaking events, or lack of exercising discretion before sharing information, etc.) or disinformation (i.e., intentional falsification of the information or distortion of facts due to disinformation campaigns, aiming to create deception, promoting an ideology or agenda, or false propaganda, etc.). This presents an opportunity for future explorations to examine and further develop the proposed methodology in assessing credibility of social media sources and isolate the ones that are more likely to disseminate false reports.

Acknowledgments This research is funded in part by the National Science Foundation's Social Computational Systems (SoCS) and Human-Centered Computing (HCC) research programs within the Directorate for Computer &Information Science &Engineering's (CISE) Division of Information &Intelligent Systems (IIS) (Award Numbers: IIS-1110868 and IIS-1110649) and the US Office of Naval Research (Grant numbers: N000141010091 and N000141410489). We would like to thank the Advances in Social Network Analysis and Mining (ASONAM) 2013 conference chairs for inviting us to develop our research further and submit the research to this publication. We are also grateful to the anonymous reviewers for their invaluable comments.

References

1. Adamic L et al (2000) Power-law distribution of the world wide web. Science 287(5461):2115–2115
2. Agarwal N, Lim M, Wigand RT (2011) Finding her master's voice: the power of collective action among female muslim bloggers. In: ECIS
3. Agarwal N, Lim M, Wigand RT (2012) Online collective action and the role of social media in mobilizing opinions: a case study on women's right-to-drive campaigns in Saudi Arabia. Web 2.0 technologies and democratic governance. Springer, New York, pp 99–123
4. Agarwal N, Lim M, Wigand RT (2012) Raising and rising voices in social media. Bus Inf Syst Eng 4(3):113–126
5. Agarwal N, Liu H, Tang L, Yu PS (2008) Identifying the influential bloggers in a community. In: Proceedings of the international conference on web search and data mining (WSDM). ACM, pp 207–218
6. Anderson C (2008) Long tail, the, revised and updated edition: why the future of business is selling less of more. Hyperion
7. Becker H, Naaman M, Gravano L (2011) Selecting quality twitter content for events. In: ICWSM, p 11
8. Bian J, Liu Y, Zhou D, Agichtein E, Zha H (2009) Learning to recognize reliable users and content in social media with coupled mutual reinforcement. In: Proceedings of the 18th international conference on world wide web. ACM, pp 51–60
9. Brin S et al (1998) The anatomy of a large-scale hypertextual web search engine. Comput Netw ISDN Syst 30(1):107–117
10. Diakopoulos N et al (2012) Finding and assessing social media information sources in the context of journalism. In: Proceedings of the 2012 ACM annual conference on human factors in computing systems. ACM, pp 2451–2460
11. Ekdale B et al (2007) From expression to influence: understanding the change in blogger motivations over the blogspan. AEJMC, Washington
12. Erkan G et al (2004) Lexrank: graph-based lexical centrality as salience in text summarization. J Artif Intell Res (JAIR) 22:457–479
13. Golub G et al (1996) Matrix computations. Johns Hopkins University Press, Baltimore
14. Gupta M, Zhao P, Han J (2012) Evaluating event credibility on twitter. In: SDM. SIAM, pp 153–164
15. Hamdy N et al (2012) Framing the Egyptian uprising in Arabic language newspapers and social media. J Commun 62(2):195–211
16. Harb Z (2011) Arab revolutions and the social media effect. M/C J 14(2):1–6
17. Haveliwala T (2003) Topic-sensitive pagerank: a context-sensitive ranking algorithm for web search. IEEE Trans Knowl Data Eng 15(4):784–796
18. Jadhav A, Purohit H, Kapanipathi P, Ananthram P, Ranabahu A, Nguyen V, Mendes PN, Smith AG, Cooney M, Sheth A (2010) Twitris 2.0: semantically empowered system for understanding perceptions from social data. Semant Web Chall
19. Johnson T et al (2004) Wag the blog: how reliance on traditional media and the internet influence credibility perceptions of weblogs among blog users. J Mass Commun Q 81(3):622–642
20. Jurczyk P, Agichtein E (2007) Hits on question answer portals: exploration of link analysis for author ranking. In: Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval. ACM, pp 845–846
21. Kleinberg J (1999) Authoritative sources in a hyperlinked environment. J ACM (JACM) 46(5):604–632
22. Kumar S, Barbier G, Abbasi MA, Liu H (2011) Tweettracker: an analysis tool for humanitarian and disaster relief. In: ICWSM
23. Langville A et al (2004) Deeper inside pagerank. Internet Math 1(3):335–380
24. Liu L, Sun L, Rui Y, Shi Y, Yang S (2008) Web video topic discovery and tracking via bipartite graph reinforcement model. In: Proceedings of the 17th international conference on world wide web. ACM, pp 1009–1018

25. LOmariba (2009) Is new media posing a serious challenge to traditional media? Technical report, University of Westminster
26. Mahata D, Agarwal N (2012) What does everybody know? Identifying event-specific sources from social media. In: CASoN, pp 63–68
27. Mahata D, Agarwal N (2013) Learning from the crowd: an evolutionary mutual reinforcement model for analyzing events. In: Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining. ACM, pp 474–478
28. Marcus A et al (2011) Twitinfo: aggregating and visualizing microblogs for event exploration. In: Proceedings of the 2011 annual conference on human factors in computing systems. ACM, pp 227–236
29. Morstatter F, Kumar S, Liu H, Maciejewski R (2013) Understanding twitter data with tweetxplorer. In: Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 1482–1485
30. Ramos J (2003) Using tf-idf to determine word relevance in document queries. In: Proceedings of the first instructional conference on machine learning
31. Rattenbury T et al (2007) Towards automatic extraction of event and place semantics from flickr tags. In: Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval. ACM, pp 103–110
32. Reese S et al (2007) Mapping the blogosphere professional and citizen-based media in the global news arena. Journalism 8(3):235–261
33. 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. ACM, pp 851–860
34. 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. ACM, pp 42–51
35. Singh V et al (2010) Mining the blogosphere from a socio-political perspective. In: 2010 international conference on Computer information systems and industrial management applications (CISIM), IEEE, pp 365–370
36. Troncy R et al (2010) Linking events with media. In: Proceedings of the 6th international conference on semantic systems. ACM, p 42
37. Tumasjan A, Sprenger TO, Sandner PG, Welpe IM (2010) Predicting elections with Twitter: what 140 characters reveal about political sentiment. ICWSM 10:178–185

Demographic and Psychographic Estimation of Twitter Users Using Social Structures

Jun Ito, Kyosuke Nishida, Takahide Hoshide, Hiroyuki Toda and Tadasu Uchiyama

Abstract Word-of-mouth marketing on social media has become more urgent with the increasing number of users and posts, and it is important to estimate user attributes because most users on Twitter do not reveal their attributes. We propose new methods for estimating user attributes of a Twitter user from the user's contents (a profile document and tweets) and social neighbors, i.e. those with whom the user has mentioned. This study has three contributions on the task of user attribute estimation. First, we investigate a labeling method that finds the users associated with a blog account and uses their profile attributes on blog as true labels of training tweet data. We confirm that using the blog labels achieved higher accuracy than manual labeling and pattern matching methods, with respect to four attributes (gender, age, occupation, and interests). Second, we validate the best way to combine bag-of-words features of profile documents and tweets. We evaluate nine combining methods and show that words in profile documents should be treated distinctively from those in tweets. Third, we reveal that to adjust amount of information from social neighbors affects estimation accuracy. We experiment three adjustment levels and show that our method, which utilizes the target user's profile document and tweets and the neighbors' profile documents (not including tweets), achieved the best accuracy. Overall experiments conducted on the estimation of the four attributes show that our method achieved higher accuracy than conventional methods that use manually-labeled tweets.

Keywords User attribute estimation · Twitter · Microblog · Social media · Blog

J. Ito (✉), T. Hoshide, H. Toda · T. Uchiyama
NTT Service Evolution Laboratories, NTT Corporation, 1-1 Hikari-no-oka, Yokosuka-shi, Kanagawa 239-0847, Japan
e-mail: ito.jun@lab.ntt.co.jp

K. Nishida
NTT Resonant Inc, 4-1-8F Granpark Tower, Shibaura 3-chome Minato-ku, Tokyo 108-0023, Japan
e-mail: k-nishi@nttr.co.jp

J. Kawash (ed.), *Online Social Media Analysis and Visualization*,
Lecture Notes in Social Networks, DOI 10.1007/978-3-319-13590-8_2

1 Introduction

Social media represented by Facebook[1] and Twitter[2] has grown rapidly in the last few years. The number of monthly active users on Facebook was more than one billion as of October 2012,[3] while Twitter had more than 200 million as of March 2013.[4] In addition, Facebook had 3.2 billion likes and comments per day and 300 million new photos daily in March 2012,[5] and Twitter had 400 million tweets per day in March 2013 (See footnote 4). Social media has attracted an unbelievable number of users who post many of their daily actions, events, and communication exchanges.

Word-of-mouth marketing on social media has become crucial with the increasing number of users and posts [9, 16]. Many opinions and impressions about products or services are among the social media posts. So social media marketing is attracting a lot of attention. The most common way to understand consumers' opinions and impressions about products or services is a questionnaire survey. However, monetary costs can be high depending on the number of monitors and questions, and real time response is not possible because it takes a long time to complete the investigation and summarize the findings. On the other hand, word-of-mouth marketing on social media can investigate many opinions and impressions at low monetary cost and in real time. Because of these benefits, many companies are working on word-of-mouth marketing tools on social media, and various marketing applications such as Radian6, Sysomos, and Forsight, are on the market.

Although word-of-mouth marketing on social media has benefits in terms of monetary cost and rapid response, its key disadvantage that it is hard to acquire user attributes. Opinions and impressions vary according to both demographic (gender, age, occupation, etc.) and psychographic (interests, etc.) attributes, so marketers analyze opinions and impressions for each attribute or examine the distribution of attributes. Questionnaire surveys can specifically ask for the user attributes, but this is not easy with social media because most users do not specify their attributes in the user profile description.

To resolve this issue, we address user attribute estimation on Twitter, which has many users and posts and the data are publicly available. We estimate four attributes (gender, age, occupation, and interests) of a Twitter user from the contents (a profile document and tweets) generated by the user and the user's social neighbors, i.e. those with whom the user has conversed (mentioned).

Contributions. (i) We investigate a labeling method that finds the users associated with a blog account and uses their profile attributes on blog as true labels of training

[1] http://www.facebook.com/.

[2] https://twitter.com/.

[3] http://newsroom.fb.com/Key-Facts.

[4] http://blog.twitter.com/2013/03/celebrating-twitter7.html.

[5] http://mashable.com/2012/04/23/facebook-now-has-901-million-users/.

tweet data. This idea was conceived by Burger et al. [2], and they demonstrated the effectiveness in gender attribute and investigated the validity by using 1,000 random sampled users. On the other hand, we confirm that using the blog labels achieved higher accuracy than manual labeling and pattern matching methods, with respect to four attributes (gender, age, occupation, and interests). (ii) We validate the best way to combine bag-of-words features of profile documents and tweets. Profile document is a high quality source of information since it may hold actual user attributes, so estimation accuracy can be increased by using it together with tweets as the training data. We evaluate nine combining methods and show that words in profile documents should be treated distinctively from those in tweets. (iii) We reveal that to adjust amount of information from social neighbors affects estimation accuracy. It is known that "birds of a feather flock together" relationships exist in social media [5, 19], however it is not clear how much information about the neighbors should be used together with those of the target user. We experiment three adjustment levels and show that our method, which utilizes the target user's profile document and tweets and the neighbors' profile documents (not including tweets), achieved the best accuracy.

Outline. Section 2 describes related work on estimating user attributes. Section 3 presents the results of our analysis of Twitter data and reveals the necessity of user attribute estimation. Section 4 details our proposed methods. Section 5 presents the results of our experiments. Section 6 summarizes the contributions of this study.

2 Related Work

We present related work on the estimation of user attributes from a user's contents and social graph. We then describe the scope of this study and the advances over existing methods.

2.1 Estimation of User Attributes from the Contents

This section details recent studies that focused on contents, i.e. a profile document and tweets. Ikeda et al. [8] extracted bag-of-words features using the Akaike Information Criterion (AIC) and estimated user attributes by Support Vector Machines (SVM). Their experiments tackled three attributes (gender, age, and location), and the results showed 88 % accuracy with regard to gender. Ikeda et al. method is used as a baseline in Sect. 5.1. The differences between their method and our method are a labeling method (manual or automatic) and an estimator (SVM or logistic regression), but an estimator is aligned with logistic regression in the experiment of Sect. 5.1 to compare the difference of a labeling method. Rao et al. [15] used n-grams or sociolinguistic features and estimated four user attributes (gender, age, location, and political orientation) by SVM. Their proposed method achieved 70–80 % accuracy in estimating

the attributes, and they reported that Twitter-specific features (number of followers, number of friends,[6] friends/followers ratio, reply ratio, number of tweets, and number of retweets[7]) are not useful. Cheng et al. [3] proposed two methods for estimating the city-level location of users: a probability model that is based on the correlation between a location and each word in the tweets and a grid-based neighborhood smoothing model for adjusting the estimation of the user location. Their method can estimate the location of 51 % users with an error range of 100 miles, from 100 tweets. Eisenstein et al. [6] estimated the location of users based on an idea similar to that of Cheng et al. [3] that there exists a strong correlation between a word and a particular region. It is different from Cheng et al. work in that it uses a generative model that estimates the latent topics and regions together. Burger et al. [2] estimated the gender of users by using a supervised learning method that employs the both word and character n-grams as features. They achieved 92 % accuracy by using the feature set of tweets, profile documents, screen name, and name. Their method performed better than manual estimation based on the Amazon Mechanical Turk[8] in terms of accuracy. Pennacchiotti et al. [14] estimated three attributes (the political orientation, race, and affinity for Starbucks Coffee) of users. They use the profile documents, tweeting tendency, and characteristic words in tweets as features. They update attribute-class label information by using the social graph and estimate user attributes by the Gradient Boosted Decision Trees (GDBT). Chu et al. [4] used the tweeting tendency, the contents of tweets, and the profile as features to distinguish human from bots by Linear Discriminant Analysis. Mislove et al. [13] compared Twitter user distribution with the actual population distribution with regard to gender, race, and location attributes. They showed that there is a bias in the distribution of users on Twitter.

2.2 Estimation of User Attributes from the Social Graph

This section describes recent studies on estimating user attributes by using the attributes of neighbors on social graph. Zamal et al. [19] tried various methods that combine a user's feature and the averaged value of the same feature for the user's neighbors. Zamal et al. method is used as a baseline in Sect. 5.4. Their method is quite different with our method in the respects of a labeling method, an estimator, and features. But the respect of how to use neighbors' information is the same. We compared the method which uses neighbors' tweets (Zamal et al. method) with our methods in Sect. 5.4. Mislove et al. [12] used the social graph of Facebook to estimate the user attributes such as the enrollment year and the department. They set the nodes that have the same attribute value as seed nodes and add the remaining nodes to these so as to increase a modularity-based value. Wen et al. [17, 18] estimated interests from large-scale monitored data, such as emails, instant messages,

[6] A friend is another Twitter user whom you are following.

[7] A Tweet by another user, forwarded to you by someone you follow.

[8] https://www.mturk.com/.

social bookmarks, and file-sharing data. They proposed a propagation model that is weighted by the number of communication events. He et al. [7] created a small group, called *homogeneous societies*, that reflects reality and used the Bayesian Network approach to estimate user attributes of blog. Zheleva et al. [20] tested whether user attributes can be estimated using friends' and group's information. They reported that group's information achieved higher accuracy than friends' information. From the viewpoint of privacy protection, Lindamood et al. [10] examined the impact of hiding a part of user attributes or user's friends information so as to prevent user attributes from being estimated by others.

2.3 Scope of This Study

The main purpose of this study is to construct a less human effort, low computational cost, but highly accurate estimator for practical use. For less human effort, we focus on labeling methods that do not require manual labeling of training data. For computational cost savings, we only utilize bag-of-words features, one-hop neighbors of the target user, and logistic regression models. For high accuracy, we add features of profile documents and social neighbors to existing tweets' feature. Therefore, our purpose is different from those of other studies that use manual labeling, Twitter-specific features, or propagate user attributes by using the entire social graph.

3 Analysis of the Twitter User's Profile

Twitter users can write self-introduction sentences (a profile document) in free-form text. It is not necessary to estimate the user attributes if these are specified in the profile document. Therefore, we analyzed Twitter data to determine how many users entered attributes in the profile document.

3.1 Structures of the Twitter Profile Field that We Used

Table 1 shows the structures of the Twitter profile field used in our study. *Description* and *location* are fields that are likely to hold user attributes. *Statuses_count* indicates

Table 1 Structures of the Twitter profile field that we used

Field name	Explanation
description	Field to hold self-introduction sentences
location	Field to hold location
statuses_count	Field to show total number of tweets
url	Field to hold website

the total number of tweets of a user. *Url* is a field for an external URL such as blog URL or the user's website's URL.

3.2 Analytical Methods and Results

Table 2 details the data analyzed. We collected over 4.6 million Japanese Twitter users who have posted at least one tweet during March 2012. We analyzed how many users filled in the *description* and the *location* fields and whether the entered text contained words that indicate user attributes. Attribute words expected were manually predefined and regular expressions were used for matching. Additionally, we calculated the description rate for each attribute.

Table 3 shows the results of the analysis. Over 82 % of users entered text in the *description*. Unfortunately, the usage of specific words was low; the description rate for age was only 3.34 %. This description rate is overly optimistic because of the presence of extraction noise. For example, age of the user's children or pets tend to be described in the user's profile document and this data must be seen as noise. Although the *location* is a special field to describe the user's location attribute and its description rate is relatively high, extraction noise is present, e.g. multiple descriptions of location that a user lived in the past.

From this analysis, we found that most users write something in their profile document but rarely their attributes. This makes it essential to estimate the user attributes. We also found that manual checking of the extraction results is inevitable because applying text matching method to a free-form text tends to yield a lot of noise.

We then counted the number of *statuses_count* and analyzed how many tweets were, on average, available as an estimation resource. The mean value was 68.1 and the median value was 553. A tweet is shorter than a blog article because it is

Table 2 The data used to analyze Twitter user's profile

Field name	Value
API level	Gardenhose (10 % random sampling)
Data collection period	3/1/2012–3/31/2012
No. of unique Japanese users	4,638,441

Table 3 Description rate of user attributes

Field name	No. of described users	Description rate (%)
Any description	3,827,885	82.53
Gender	353,558	7.62
Age	154,900	3.34
Occupation	631,626	13.62
Location	1,158,570	24.98

limited to 140 characters, and the mean value of *statuses_count* is low. Therefore, the estimation accuracy may decrease if the target user does not have enough number of tweets.

4 Proposed Methods

In the following sections, we propose a labeling method, content-based methods, and social-graph-based methods to solve the problems described in Sect. 3.

4.1 Labeling Method

It is difficult and time-consuming to label training data manually. Although we can use regular expressions to label automatically, this method contains extraction noise as shown in Sect. 3, so manual confirmation is needed. The reason why labeling fails so often is that profile documents hold free-form text. On the other hand, blog's profile fields are usually provided for each attribute and multiple choice selections are provided for user entry. It is easy to extract user attributes from these fields by using a rule-based method. Therefore, automatic learning is available by using the users who have both blog and Twitter accounts, we call them TwiBlo users, as training data.

We counted up the number of distinct URLs in the *url* field for each domain from the users shown in Table 2. Top 10 results are showed in Table 4. To see the results, the 1st rank domain has about 160 thousand TwiBlo users, and the top 10 blog sites have over 240 thousand TwiBlo users. This is 10–100 times higher than the amount of manually labeled data [8, 14, 15]. Manual labeling is difficult and time-consuming, so the amount of data tends to be less than several thousand. In general, estimation accuracy increases with the amount of training data, so this labeling method is expected to achieve higher accuracy than manual labeling methods.

Here, we explain the flow of this labeling method. It first finds users who have both Twitter and blog accounts (TwiBlo users) by using the URL described in the *url* field. It then extracts the attributes that a user specified in his/her blog as true labels of the training data in Twitter about the user. The extraction rules are predefined for each blog site and extraction script outputs user attributes when it is input the URL described in the *url* field.

Table 4 Domain names present in the url field

Rank	Domain name	No. of users	Rank	Domain name	No. of users
1	ameblo.jp	159,768	6	d.hatena.ne.jp	12,348
2	blog*.fc2.com	20,407	7	jugem.jp	11,991
3	facebook.com	20,237	8	blogspot.com	11,752
4	blog.livedoor.jp	16,500	9	exblog.jp	10,706
5	mixi.jp	16,289	10	tumblr.com	10,647

Burger et al. [2] also used TwiBlo users as training data, but they verified the effectiveness of this method only in gender and conducted validity check manually by using one thousand randomly sampled users. By contrast, we demonstrate the effectiveness of this method with respect to four attributes and conduct verification experiments in Sects. 5.1 and 5.2.

4.2 Content-Based Methods

We lack enough information to estimate user attributes depending on the target user's tweeting history when only his/her tweets are used as training data. Accurate estimation requires more information than is contained within the tweets, so we consider the use of profile documents together with tweets. There is only one profile document per user, but it is a high quality source of information since it may hold actual user attributes, so estimation accuracy can be increased by using it together with tweets as the training data. However, we do not know what is the best way to mix/combine profile documents and tweets. Therefore, we create following nine methods and determine which is best in Sect. 5.

MIX. This estimator counts words from profile documents and tweets without distinction; this is equal to consider a profile document as a tweet.
JOIN. This estimator treats words from profile documents differently from those in tweets. Thus, the number of feature dimension is greater than the estimator using only tweets.
AVG. Creates two estimators, one each from profile documents and tweets. The outputs of the estimators are averaged.
MAX. Creates two estimators, one each from profile documents and tweets. The output with maximum value is adopted.
VAR. Creates two estimators, one each from profile documents and tweets. The output with maximum variance is adopted.
DEF. Creates two estimators, one each from profile documents and tweets. The ratio of margin of defeat between 1st and 2nd classes is calculated for each estimator's output and the output with the minimum ratio is adopted.
KIND. Creates two estimators, one each from profile documents and tweets. The equations are as follows:

$$P(u) = R_p(u)P_p(u_p) + R_t(u)P_t(u_t), \tag{1}$$

$$R_p(u) = \frac{I_t(u_t)}{I_p(u_p) + I_t(u_t)},$$

$$R_t(u) = \frac{I_p(u_p)}{I_p(u_p) + I_t(u_t)},$$

$$I_p(u_p) = -\log\left(\frac{\mathrm{kind}(u_p) + \alpha}{|F_p|}\right), \tag{2}$$

$$I_t(u_t) = -\log\left(\frac{\text{kind}(u_t) + \alpha}{|F_t|}\right). \tag{3}$$

Let u be a user with the user's profile document u_p and tweets u_t. As indicated by Eq. (1), final estimation probability P is obtained by aggregation of estimation probabilities from the profile document P_p and tweets P_t weighted by reliability scores R_p and R_t, respectively. R_p and R_t are calculated by self-information I_p and I_t that are obtained by the log ratio of the kind function's value and the total number of kinds of features $|F_p|$ and $|F_t|$ shown as Eqs. (2) and (3), respectively. F_p and F_t are extracted in a feature selection step; in this study, we employ the Akaike Information Criterion (AIC) [1] as the feature selection method [8, 11] for all the content-based methods. The kind function returns the number of kinds of features used to estimate the target user's attributes. α is a constant value to prevent the logarithm from being zero; we set it to 1.

AIC. Creates two estimators, one each from profile documents and tweets. Equations (2) and (3) in **KIND** are replaced with (4) and (5), respectively:

$$I_p(u_p) = -\log\left(\frac{\sum_{s\in\text{set}(u_p)} \text{aic}(s) + \alpha}{\sum_{f\in F_p} \text{aic}(f)}\right), \tag{4}$$

$$I_t(u_t) = -\log\left(\frac{\sum_{s\in\text{set}(u_t)} \text{aic}(s) + \alpha}{\sum_{f\in F_t} \text{aic}(f)}\right), \tag{5}$$

where set is a function that returns the set of features contained in the input text, and aic is another function that returns the AIC value of the input feature f, which is calculated in the feature selection step.

RANK. Creates two estimators, one each from profile documents and tweets. Equations (2) and (3) in **KIND** are replaced with (6) and (7), respectively:

$$I_p(u_p) = -\log\left(\frac{\sum_{s\in\text{set}(u_p)} \text{rank}(s) + \alpha}{\sum_{f\in F_p} \text{rank}(f)}\right), \tag{6}$$

$$I_t(u_t) = -\log\left(\frac{\sum_{s\in\text{set}(u_t)} \text{rank}(s) + \alpha}{\sum_{f\in F_t} \text{rank}(f)}\right), \tag{7}$$

$$\text{rank}(f) = \frac{|F|}{\text{index}(f)}, \tag{8}$$

where rank is a function that returns the rank value of the input feature $f \in F$ (F would be F_p or F_t) shown as Eq. (8). The rank value is obtained by the ratio of the index function's value and the total number of kinds of features. The index function returns the index of features sorted by AIC values, which are calculated in the feature selection step, in descending order; initial value is 1.

4.3 Social-Graph-Based Methods

Neighborhood users on social graph are known to tend to have similar attributes; this is called the "birds of a feather flock together" effect [5]. Utilizing the information about neighbors can make user attribute estimation more stable even if the user does not have enough information to permit his/her user attributes to be estimated.

We propose new methods that use the information of the target user's neighbors by extending Zamal et al. methods [19]. The main difference of our methods is the information used from the target user and the social neighbors. We set three information levels as follows.

PR. Uses profile documents only (PRofile).
TW. Uses tweets only (TWeets).
TP. Uses both profile documents and tweets (Tweets and Profile).

We tried various combinations of these information levels between a target user and his/her social neighbors. Here, Zamal et al. method is equivalent to the case of applying TW to both the target user and social neighbors. However, it is not exactly the same in terms of features used; they used various features such as n-grams or tweeting tendency, but we use only bag-of-words features.

The methods of selecting social neighbors and combining those features with the target user's features are the same as in Zamal et al. as follows.

ALL. All social neighbors are used.
MOST. N-most popular social neighbors are selected. The popularity is assessed in terms of the number of followers of the user. We set 10 for N in our study (following N is the same value).
LEAST. N-least popular social neighbors are selected. It is based on an expectation that a user dares to follow unpopular users because they are real friends.
CLOSEST. N-most closest social neighbors are selected. Closeness is determined by the number of conversations (mentions) from a target user to the neighbors.

The features of the selected neighbors are averaged and then combined with the target user's feature. AVG and JOIN are employed as the combination methods. AVG averages the feature sets of the target user and the social neighbors, and JOIN treats those sets distinctively and joins them.

There are various ways to construct a social graph, e.g. using the relation of friends, followers, and mentions. We use mention relations because of its characteristics and Twitter API's restrictions. It is a closer relationship than friends/followers relationships because mention is an active action. Twitter provides REST API which returns the friends/followers of a user, but its rate limit is 350 calls per hour now, and will fall to 15 calls per 15 min from May 2013. It is hard to get friends/follower networks given this API restriction. On the other hand, mentioning relationship is extracted from tweets collected by the streaming API, so it is not necessary to use the REST API.

5 Evaluation Experiments

We conducted experiments to evaluate the effectiveness of our proposed methods. Table 5 details the data used in the evaluation experiments, and Table 6 shows classes in each attribute. We crawled about 86 thousand TwiBlo users, who are Twitter users associated with a blog account, and up to 200 tweets, excluding RTs, per user. Blog users and their articles are different from TwiBlo user's data and this blog data are used in Sect. 5.2 for evaluating our labeling method. We used AIC for the feature selection method and LIBLINEAR[9] with L2-regularized logistic regression (primal) for constructing the estimators. The reason we used logistic regression is to get probabilistic output. Based on the results of a preliminary experiment, we set the number of features and the cost parameter, which is C parameter for L2-regularized logistic regression (primal). Five thousand and thirty thousand features were extracted from profile documents and tweets, respectively. The cost parameter was set to 1.

Table 5 Experimental data

	Field name	TwiBlo users[a]	Blog users
	Gender	71,129	49,739
	Age	36,234	17,689
No. of users	Occupation	41,920	37,427
	Interests	20,846	7,417
	Total	86,183	65,873
No. of blog articles		796,583	626,903
No. of tweets		15,124,094	–

[a] Twitter users associated with a blog account

Table 6 Classes in each attribute

Attribute	Classes (number of class)
Gender	Male and female (2 classes)
Age	10s, 20s, 30s, and over 40s (4 classes)
Occupation	Company employee, government employee, self-employed, job-hopping part-time worker, housewife, graduate or undergraduate student, middle or high-school student, and others (8 classes)
Interests	Reading, cooking, traveling, gourmets, sweets, computers, internets, games, musics, sports, pets, comics and animations, television and movies, politics and economics, learning and education, health maintenance, career development, finance, entertainment world, and fashion (20 classes)

[9] http://www.csie.ntu.edu.tw/~cjlin/liblinear/.

5.1 Comparison Against Manual Labeling

We conducted an experiment to compare the estimation accuracy yielded by our labeling methods and three other labeling methods. REGEXP is a labeling method that matches specific words which associate user attributes to words contained in profile documents by using regular expressions. HUMAN is another labeling method that first author surveys REGEXP's output and retains only correct results; it corresponds to Ikeda et al. method [8]. D1000 is the same as our labeling method (DIRECT), but its data size is equal to REGEXP and HUMAN.

We targeted gender and age attributes and based the evaluation on special 5-fold cross-validation. Figure 1 shows an overview of the experiment. TwiBlo users' data were divided into five blocks, one was used as test data for all four methods, so all methods evaluated the same data. On the other hand, the training data were different for each of the four methods. DIRECT used the rest of the data to construct an estimator (model) when TwiBlo users' data were folded, and D1000 used the same data but its size was restricted to 1,000 for each class by random sampling. REGEXP and HUMAN used the data obtained by each method and their size was 1,000 for each class. The same operation was repeated for the number of splits.

The results are shown in Table 7. The estimation accuracy of both gender and age attributes rises gradually in the order of REGEXP, HUMAN, D1000, and DIRECT. Though the amount of data was the same, D1000 was more accurate than HUMAN. It is because HUMAN used minority and special users who specified their attributes

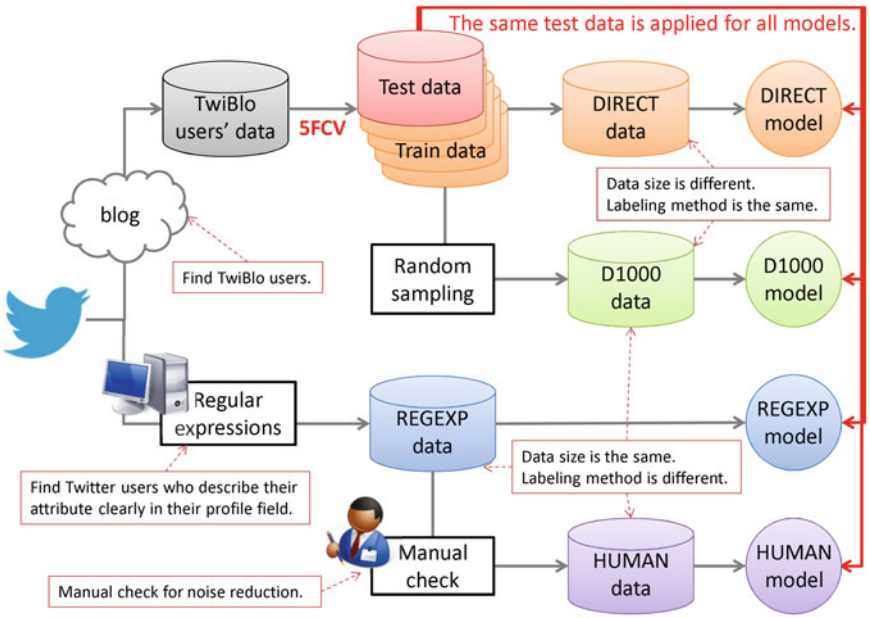


Fig. 1 An overview of the experiment in Sect. 5.1

Table 7 Accuracy with each labeling method

	REGEXP	HUMAN	D1000	DIRECT
Gender	72.59	82.32	89.39	**94.50**
Age	59.49	61.86	67.72	**76.28**

in their profile document. DIRECT achieved the best accuracy because its training data size was 35 and 9 times larger than the others in regard to gender and age, respectively. While it is difficult and time-consuming to collect training data manually, our labeling method can collect large amounts of training data automatically. Therefore, our labeling method can automatically make a more accurate estimator than the manual-based method.

5.2 Validation of Using Labels from Blog

Our labeling method learns from user tweets and the true labels found in his/her blog's profile. When a user lies about his/her attributes in blog or writes completely different content between Twitter and blog, learning does not go well. For example, a user specifies travel as his interest attribute in his blog's profile, but uses Twitter as a communication tool and his tweets contain no mention of travel.

We tried to remove from the training data users who entered different content in Twitter and blog. We tested two types of filtering method. The first used an estimator learned from randomly collected blog articles. It estimates user attributes from his/her tweets and blog articles, and filters out users whose estimation results differ from the blog's profile. BOTH is the case that all three factors (estimation results from his/her tweets and blog articles, and his/her blog's profile) are the same. BLOG and TWIT are the case that the estimation results from blog articles and tweets are the same as the blog profile and the other factors are different from the blog profile, respectively. These filtering methods depend on the accuracy of the estimator, so the second filtering method uses cosine similarity (COS) rather than an estimator. Bag-of-words word frequency vectors were created from his/her tweets and blog articles, and users whose cosine similarity between both vectors was less than 0.8 were dropped. We also evaluated the transfer learning method (TRANS) and the non-filtering method (DIRECT) which equals our labeling method. TRANS uses blog articles and tweets as training data and test data, respectively.

We conducted experiments based on special 5-fold cross-validation. Figure 2 shows an overview of the experiment. We divided TwiBlo users' data into five blocks, one was used as the test data, and the remaining data were filtered by the methods described above; the filtered results were used as training data. DIRECT is not filtered and TRANS's training data are different from the other methods, but all methods had the same test data. The same operation was repeated for the number of splits.

The results are shown in Table 8. Accuracy is shown as percentage and the amount of training data after filtering is given by the bracketed number. DIRECT achieved the best accuracy in all attributes except for interests. BLOG and COS achieved the

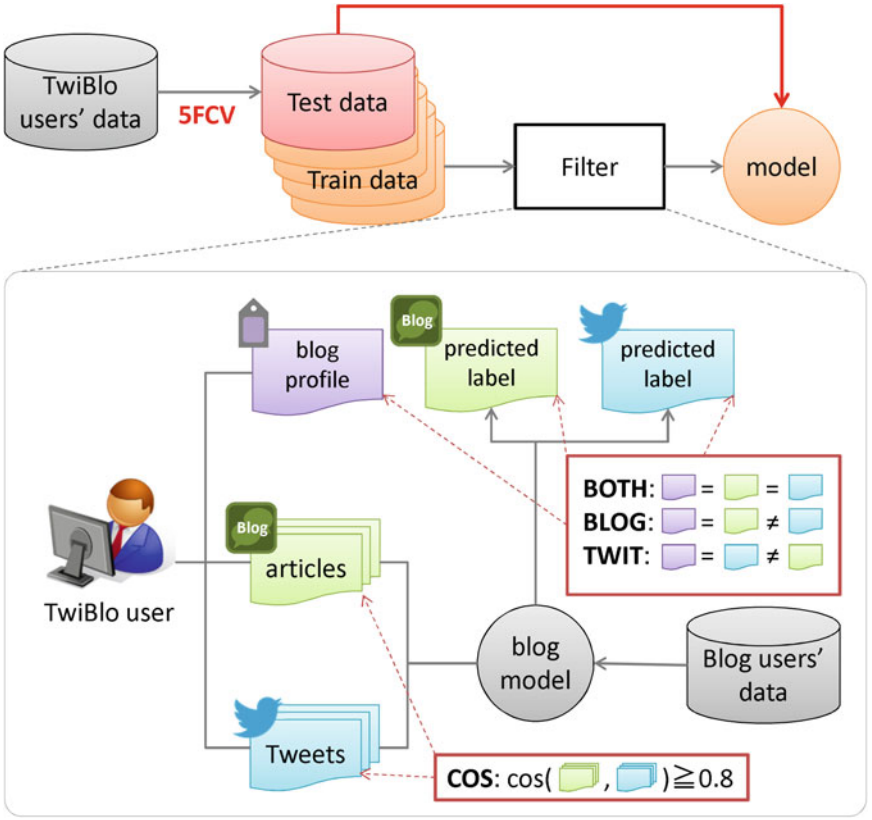


Fig. 2 An overview of the experiment in Sect. 5.2

Table 8 Accuracy with each filtering method

	DIRECT	BOTH	BLOG	TWIT	COS	TRANS
Gender	**94.37** (71,129)	90.58 (55,728)	93.43 (62,291)	91.12 (60,929)	93.24 (29,873)	85.66
Age	**75.82** (36,234)	68.01 (17,661)	71.60 (23,569)	68.76 (23,964)	70.92 (14,751)	66.14
Occupation	**62.29** (41,920)	50.66 (14,382)	55.16 (20,489)	52.35 (20,883)	56.69 (16,912)	49.82
Interests	55.35 (22,393)	49.82 (7,960)	**55.38** (12,851)	50.96 (10,457)	**55.38** (9,222)	42.32

best accuracy in interests attribute, but the difference with DIRECT, 0.03 %, was not statistically significant in McNemar's test. COS creates a small training data set, but its accuracy does not decrease commensurately. TRANS failed to achieve good accuracy because test and training data had different domains.

From these results, we conclude that falsification of blog profile entries and the difference in content between Twitter and blog is little. Therefore, we are able to apply our labeling method without filtering the training data, and this achieves high accuracy for all attributes examined here.

5.3 Comparison of Methods that Use Profile Documents

To confirm that profile documents must be used together with tweets to improve the estimation accuracy, we conducted experiments on the various methods shown in Sect. 4.2. In addition to these methods, we also evaluated the method that uses only profile documents (PROF) and the method that uses tweets only (TWEET). TwiBlo users' data were used for the experiments, and four attributes (gender, age, occupation, and interests) were evaluated with 5-fold cross-validation.

The results are shown in Table 9 and Figs. 3, 4, 5, and 6. The table lists estimation accuracies for each method and attribute, and the average of the ranks for each method. Figures plots accuracy–coverage curves (ACC); x-axis is coverage and y-axis is accuracy. The best possible prediction method would yield points in the upper

Table 9 Accuracy comparison of methods that use profile documents

	PROF	TWEET	MIX	JOIN	AVG	MAX	VAR	DEF	KIND	AIC	RANK
Gender	78.98	94.20	94.20	94.46	94.03	94.03	94.03	94.03	94.46	94.53	**94.60**
Age	60.43	72.26	72.90	**73.91**	73.20	72.95	72.89	72.63	73.43	73.45	73.36
Occupation	52.29	58.71	58.84	61.30	**61.81**	61.13	60.74	61.21	61.49	61.45	61.46
Interests	54.00	56.92	57.73	**61.56**	60.51	59.88	59.55	60.23	60.29	60.38	60.18
AOR[a]	11	9	7.5	**2.75**	4.125	7.125	8.125	7.125	3	**2.75**	3.5

[a] Average of the ranks

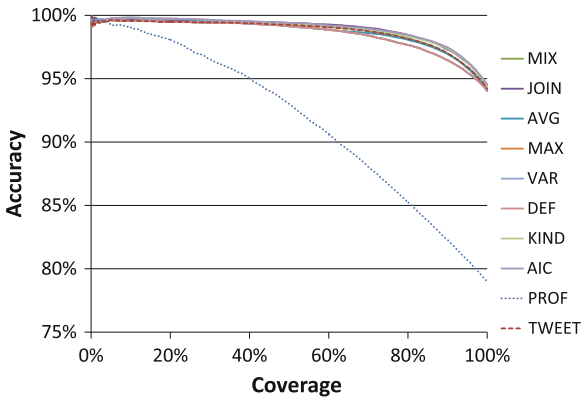


Fig. 3 Accuracy–coverage curve of gender

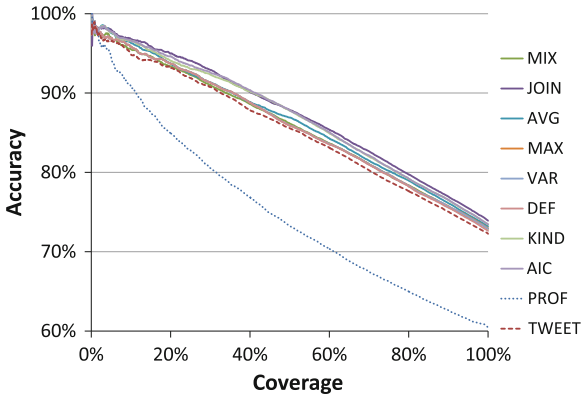


Fig. 4 Accuracy–coverage curve of age

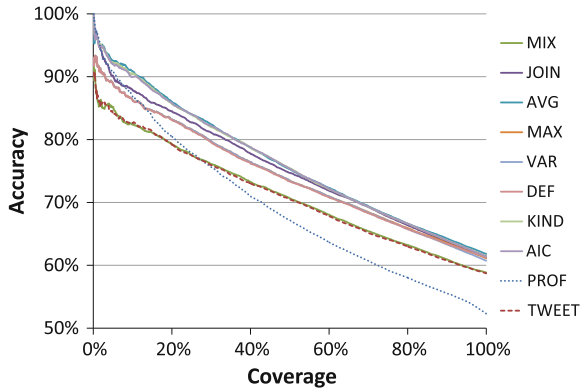


Fig. 5 Accuracy–coverage curve of occupation

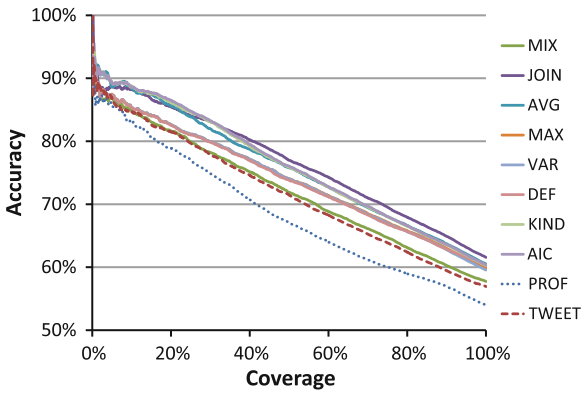


Fig. 6 Accuracy–coverage curve of interests

side of ACC space; it means always 100 % accuracy in any coverage. ACC is good to be plotted near the top right corner.

JOIN and AIC achieved the best average of the ranks as shown in Table 9. JOIN achieved the best accuracy in gender and interests attributes, and AIC offered high and stable accuracy but not the highest accuracy. With both methods, a significant difference was observed in McNemar's test at a significance level of 5 % compared to TWEET in all attributes. Therefore, we propose using JOIN or AIC method, either of which use profile documents together with tweets, to estimate user attributes. JOIN achieved higher accuracy than MIX, so words in profile documents should be treated differently from words in tweets.

The table shows the accuracy only at the point of 100 % coverage, but the figures allow us to grasp more details. The figures show that the accuracy tends to decrease monotonically, and the accuracy ranking of methods remains basically unchanged as coverage changes. The difference in the accuracies of the methods appears most strongly in the attributes that have many classes, i.e. occupation (8 classes) and interests (20 classes). PROF has higher accuracy than TWEET at low coverage levels as shown in Fig. 5. This indicates that the number of users who specify helpful words to estimate occupational attribute in their profile document is larger than the other attributes. This result presents that the effect of using profile documents varies according to attributes.

5.4 Evaluation of Social-Graph-Based Method

We conducted experiments on the methods shown in Sect. 4.3, and the results are shown in Table 10. The methods tested are written as [the combination method of feature sets (AVG or JOIN)]-[the target user's information type (TW or TP)][the neighbors' information type (PR, TW, or TP)]-[the selection method of the neighbors (ALL, MOST, LEAST, or CLOSEST)], e.g. AVG-TWTW-ALL. We also evaluated the methods that use the target user's profile document only (PROF), tweets only (TWEET), both the profile document and tweets (TWEPRO; it is the same as JOIN in Sect. 4.2), and the neighbors' tweets only (NBR).

We based experiments on 10-fold cross-validation, and the table shows accuracies for each method and user attribute. Accuracy values higher than TWEPRO, which is a baseline method that does not use neighbors' information, are bold faced to simplify comparison with the content-based method, and the highest accuracy for each attribute is indicated by an asterisk. The experimental data are the same as shown in Table 5, but size was confined to 200 for each class due to the difficulty of collecting large numbers of neighbors' tweets. So, accuracies are low compared to Table 9 because of the reduction of training data.

Table 10 indicates that almost social-graph-based methods yielded higher accuracy than TWEPRO (content-based method) for age attribute, and NBR yielded lower accuracy than TWEPRO for all attributes except age. These indicate that the

Table 10 Accuracy comparison of each social-graph-based method

	Gender	Age	Occupation	Interests
PROF	65.78	47.90	36.10	40.03
TWEET	85.25	61.38	46.68	47.53
TWEPRO	85.75	61.88	47.51	53.49
NBR-ALL	69.87	**63.28**	42.16	40.69
NBR-MOST	68.57	58.33	37.18	37.01
NBR-LEAST	70.13	61.59	41.43	37.34
NBR-CLOSEST	68.31	59.24	39.58	37.06
AVG-TWTW-ALL	74.25	**64.38**	44.01	44.45
AVG-TWTW-MOST	72.25	60.75	41.26	42.49
AVG-TWTW-LEAST	73.00	**64.13**	44.58	43.22
AVG-TWTW-CLOSEST	71.50	61.38	41.52	43.54
JOIN-TWTW-ALL	83.00	**64.88**	47.45	48.00
JOIN-TWTW-MOST	80.25	**62.13**	45.03	46.98
JOIN-TWTW-LEAST	83.00	**63.63**	47.32	47.01
JOIN-TWTW-CLOSEST	79.25	**64.00**	45.92	47.58
JOIN-TPPR-ALL	**86.75**	**65.00**	**48.09**	**54.45***
JOIN-TPPR-MOST	**86.75**	**63.88**	**48.21**	**54.27**
JOIN-TPPR-LEAST	**86.25**	**64.63**	**48.41**	**53.64**
JOIN-TPPR-CLOSEST	**87.25***	**64.25**	**48.66**	**53.75**
JOIN-TPTW-ALL	83.00	**65.13**	**48.72***	52.81
JOIN-TPTW-MOST	80.50	**62.50**	46.17	51.50
JOIN-TPTW-LEAST	84.25	**63.75**	**48.21**	51.50
JOIN-TPTW-CLOSEST	80.25	**63.50**	**47.64**	52.21
JOIN-TPTP-ALL	82.50	**66.63***	**48.28**	52.73
JOIN-TPTP-MOST	79.75	**62.88**	46.62	51.76
JOIN-TPTP-LEAST	83.25	**64.25**	**47.83**	51.35
JOIN-TPTP-CLOSEST	80.25	**64.00**	47.07	51.87

strength of homophily[10] depends on the attributes and that of age is strong. JOIN achieved higher (and stably) accuracy than AVG, so words from the target user and the neighbors should be treated differently. JOIN-TPPR-* is superior in accuracy to TWEPRO and Zamal et al. methods (*-TWTW-*), and achieved the best accuracy in gender and interests attributes. In general, the estimation accuracy increases with increasing the amount of training data, but the neighbors data can be a noise when the estimating attribute has weak homophily. These results showed that the estimation accuracy becomes stable for all attributes by using only information from the profile documents of the neighbors. Unfortunately, we were not able to find a clear winner among the neighbors' selection methods.

[10] The tendency of individuals to associate and bond with similar others.

6 Conclusion

We propose new methods for estimating user attributes of a Twitter user from the user's contents (a profile document and tweets) and social neighbors, i.e. those with whom the user has mentioned. Although there are many studies on estimating user attributes, we proposed new methods and showed new knowledge as indicated by the following three points. First, we investigate a labeling method that finds the users associated with a blog account (TwiBlo users) and uses their profile attributes on blog as true labels of training tweet data. We confirm that using the blog labels achieved higher accuracy than manual labeling and pattern matching methods, with respect to four attributes (gender, age, occupation, and interests). Additionally, our labeling method does not require any filtering of training data, because our experiments showed that the influence of blog's profile falsification and the difference in the Twitter and blog contents are slight. Second, we validate the best way to combine bag-of-words features of profile documents and tweets. We evaluate nine combining methods and show that words in profile documents should be treated distinctively from those in tweets. Our experimental results revealed that JOIN and AIC, described in Sect. 4.2, are the best choices. Third, we reveal that to adjust amount of information from social neighbors affects estimation accuracy. We experiment three adjustment levels and show that our method, which uses the target user's profile document and tweets and the neighbors' profile documents, experimented as JOIN-TPPR-* in Sect. 5.4, achieved the best accuracy.

We proposed a labeling method using blog as an example of a true label propagation source, however other media such as Facebook can be used instead of blog if the media holds user attributes as true labels. In future work, we will investigate the impact on estimation accuracy of using other media.

References

1. Akaike H (1974) A new look at the statistical model identification. IEEE Trans Autom Control 19(6):716–723
2. Burger JD, Henderson J, Kim G, Zarrella G (2011) Discriminating gender on Twitter. In: EMNLP, pp 1301–1309
3. Cheng Z, Caverlee J, Lee K (2010) You are where you tweet: a content-based approach to geo-locating Twitter users. In: CIKM, pp 759–768
4. Chu Z, Gianvecchio S, Wang H, Jajodia S (2010) Who is tweeting on Twitter: human, bot, or cyborg? In: ACSAC, pp 21–30
5. Conover M, Gonçalves B, Ratkiewicz J, Flammini A, Menczer F (2011) Predicting the political alignment of Twitter users. In: SocialCom, pp 192–199
6. Eisenstein J, O'Connor B, Smith NA, Xing EP (2010) A latent variable model for geographic lexical variation. In: EMNLP, pp 1277–1287
7. He J, Chu WW, Liu ZV (2006) Inferring privacy information from social networks. In: ISI, pp 154–165
8. Ikeda K, Hattori G, Matsumoto K, Ono C, Higashino T (2012) Demographic estimation of twitter users for marketing analysis. IPSJ Trans Consum Devices Syst 2(1):82–93

9. Jansen BJ, Zhang M, Sobel K, Chowdury A (2009) Twitter power: tweets as electronic word of mouth. J Am Soc Inf Sci Technol 60(11):2169–2188
10. Lindamood J, Heatherly R, Kantarcioglu M, Thuraisingham B (2009) Inferring private information using social network data. In: WWW, pp 1145–1146
11. Matsumoto K, Hashimoto K (1999) Schema design for causal law mining from incomplete database. In: DS, pp 92–102
12. Mislove A, Viswanath B, Gummadi KP, Druschel P (2010) You are who you know: inferring user profiles in online social networks. In: WSDM, pp 251–260
13. Mislove A, Lehmann S, Ahn YY, Onnela JP, Rosenquist JN (2011) Understanding the demographics of Twitter users. In: ICWSM
14. Pennacchiotti M, Popescu AM (2011) Democrats, republicans and starbucks afficionados: user classification in Twitter. In: KDD, pp 430–438
15. Rao D, Yarowsky D, Shreevats A, Gupta M (2010) Classifying latent user attributes in Twitter. In: SMUC, pp 37–44
16. Trusov M, Bucklin RE, Pauwels K (2009) Effects of word-of-mouth versus traditional marketing: findings from an internet social networking site. J Mark 73(5):90–102
17. Wen Z, Lin CY (2010) On the quality of inferring interests from social neighbors. In: KDD, pp 373–382
18. Wen Z, Lin CY (2011) Improving user interest inference from social neighbors. In: CIKM, pp 1001–1006
19. Zamal FA, Liu W, Ruths D (2012) Homophily and latent attribute inference: inferring latent attributes of Twitter users from neighbors. In: ICWSM
20. Zheleva E, Getoor L (2009) To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles. In: WWW, pp 531–540

Say It with Colors: Language-Independent Gender Classification on Twitter

Jalal S. Alowibdi, Ugo A. Buy and Philip S. Yu

Abstract Online Social Networks (OSNs) have spread at stunning speed over the past decade. They are now a part of the lives of dozens of millions of people. The onset of OSNs has stretched the traditional notion of community to include groups of people who have never met in person but communicate with each other through OSNs to share knowledge, opinions, interests and activities. Here we explore in depth language independent gender classification. Our approach predicts gender using five color-based features extracted from Twitter profiles such as the background color in a user's profile page. This is in contrast with most existing methods for gender prediction that are language dependent. Those methods use high-dimensional spaces consisting of unique words extracted from such text fields as postings, user names, and profile descriptions. Our approach is independent of the user's language, efficient, scalable, and computationally tractable, while attaining a good level of accuracy.

Keywords Color-based feature · Gender classification · Twitter profile · Color quantization · Color feature · Low dimensional space · Social network analysis · Online social network

1 Introduction

Online Social Networks (OSNs) generate a huge volume of user-originated texts. OSNs allow users to share knowledge, opinions, interests, activities, relationships and friendships with each other. Gender classification can serve multiple purposes in these

J.S. Alowibdi · U.A. Buy (✉) · P.S. Yu
Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA
e-mail: buy@uic.edu

P.S. Yu
e-mail: psyu@uic.edu

J.S. Alowibdi
Faculty of Computing and Information Technology, King Abdulaziz University,
Jeddah, Saudi Arabia
e-mail: jalowi2@uic.edu

J. Kawash (ed.), *Online Social Media Analysis and Visualization*,
Lecture Notes in Social Networks, DOI 10.1007/978-3-319-13590-8_3

settings. Commercial organizations can use gender classification for advertising. Law enforcement may use gender classification as part of legal investigations. Others may use gender information for social reasons. Here we examine gender classification based solely on color preferences. We specifically present a novel approach for predicting gender using five color-based features extracted from Twitter profile colors (e.g., the background color in a user's profile page).

Methods for gender classification are typically language dependent, not scalable, inefficient, and held offline using high-dimensional spaces. A recent study [1] shows that there are around 78 different languages in Twitter with English as the dominant language. Another study by Wauters [2] shows that only around 50 % of Twitter messages are in English. Our Twitter dataset alone contains 34 different languages. An estimate breakdown of language use in our dataset shows that around 69 % users are English speaking with the remaining 31 % distributed over 33 languages. In addition, around 20 % of the 69 % users who set their profiles to be as English speaking routinely post texts in different languages than English. Thus, about 55 % of users in our dataset use languages different than English for their posts and profiles. Our long-term goal is gender identification in OSNs with an emphasis on accuracy, computational efficiency and scalability of gender predictions. We are especially interested in language-independent methods.

To date, most existing approaches to gender classification on Twitter depend heavily on an analysis of text in posted messages, aptly called tweets; however, the strength of profile colors for gender classification is currently unknown. Most existing research for gender classification on Twitter is language dependent. An existing study for gender classification [3] shows that 66 % of users in their dataset use English. Other works for gender classification [4–6] did not mention the language distribution of their Twitter dataset, which we assume to be in English. In contrast, our dataset contains profiles of users of all ages, languages, and cultures.

To date, Burger et al. [3] used four different characteristics from a user's profile and posts (i.e., first name, user name, description and tweets) for gender classification. Liu and Ruths [7] utilized only first names for gender classification. Alowibdi et al. [8] applied a phoneme-based analysis to characteristics extracted from a user's profile (e.g., first names and user names). Other works for gender classification use user posts and other statistical information, such as friends and followers, in order to identify gender [4–6, 9]. In general, all existing approaches to gender classification on Twitter use word-based n-grams resulting in a huge feature space consisting of unique words and word combinations extracted from tweets. The size of the resulting feature sets is often in the order of many million features [3]. On the whole, our work to predict gender from profile's colors is unique and different from existing methods in term of its simplicity, language independence and low computational space and time complexity. In addition, our work is different because of the range of profile colors characteristics that we consider.

We predicted automatically the gender value of users based on their color preferences. We analyzed user profiles with different classifiers in the Konstanz Information

Miner (KNIME), which uses the Waikato Environment for Knowledge Analysis (WEKA) machine learning package [10, 11]. Unlike text-based approaches, we used a novel method for predicting gender using five color-based features. Our preliminary results with our dataset are quite encouraging. Although we are considering only five color-based features, we can predict gender with an accuracy of 74.2 %, a gain of about 24 % with respect to a 50 % baseline. A key to the success of our gender guessing with colors is our preprocessing of color features using a quantization technique that we discuss later on. An advantage of our method is its broad applicability to Twitter users regardless of their language; we use only color-based features to identify gender. In addition, our color-based analysis shows promising results in term of computational complexity compared to other gender-guessing methods, which use a much larger feature set. Our approach utilizes only five color-based features while Burger et al. [3] and Rao et al. [6] use text sentiment with 1.2 million and 15.4 million features. Our results show that colors alone can provide reasonably accurate gender predictions, even though a substantial number of users we analyzed do not change the default colors provided by Twitter in their Twitter profiles or in other web sites hosting their profiles (e.g., Twitter App). We conclude that colors are a good gender indicator for users who do change the default colors in their profiles. In these cases, we will be able to use colors alone as part of our gender classification methods.

Our main contributions are outlined below.

1. We defined a novel, language-independent approach for predicting gender using color-based features. Most other existing methods rely on text, which varies by language.
2. We validated our approach by applying different classifiers to a large dataset of Twitter profiles. Our results show that colors alone can provide reasonably accurate gender predictions. In some cases, we can predict gender with compatible accuracy of 74.2 %, a gain of about 24 % with respect to a 50 % baseline.
3. We defined a color quantization and sorting technique for preprocessing colors harvested from Twitter profiles. This technique substantially improves prediction accuracy while also reducing dramatically the size of our feature set. As a result, our color-based analysis has much lower computational complexity than most other gender-guessing methods, which use much larger feature sets based on text features.
4. We concluded that colors alone are not useful features. However, we found that considering a combination of multiple (five) color selections from each Twitter profile leads to a reasonable degree of accuracy for gender prediction.

The remainder of this paper is organized as follows. In Sect. 2, we briefly summarize related work on gender classification. In Sect. 3, we described our dataset collection. In Sect. 4, we detail our proposed approach. In Sect. 5, we report our empirical results from different classifiers and we analyze these results Finally, in Sect. 6, we give some conclusions and outline future work.

2 Related Work

Many researchers have investigated gender classification. Lexical richness measures based on word-frequencies have also been studied [12]. For instance, Argamon et al. [13] defined a Part-Of-Speech (POS) n-gram technique to capture author writing styles. Many authors have studied POS tags, unigrams, word-frequencies, word-classes, POS patterns, POS contents and POS style metrics [14–20]. Unlike those works, Burger et al. [3] and Rao et al. [6] worked on gender classification on Twitter postings by utilizing text sentiment. In particular, Rao et al. [6] use sociolinguistic-feature models, n-gram feature models and stacked models for gender classification utilizing text sentiment. Burger et al. use the n-gram feature model [3]. Both approaches generate millions of features from text sentiments.

In summary, most existing authors explore gender classification by utilizing language-based methods. Researchers in the natural language processing and data mining communities worked on gender classification of different systems including OSNs for the past several years. Despite the challenging feature set of those systems, researchers have studied various schemes for defining feature feasibility and stability. The drawback of using text sentiment is high computational complexity of the dimensional space generated, language dependency, and millions of features. Our work shows that reasonably accurate predictions are possible using only five color-based features.

3 Dataset Collection

We chose Twitter profiles as the starting point of our data collection for several reasons. First, Twitter is one of the most popular social networks to date with a huge user community cutting across great many languages, cultures and age groups. In early 2013, Twitter reached 555 million registered users [21]. As of this writing, Twitter states that there are more than 200 million active users producing around 400 million tweets per a day [22]. Second, Twitter has all the color attributes that we need to set up our experiments. These attributes are generally public, meaning that they can be accessed and viewed by anyone who requests them. Lastly, Twitter provides a rich Application Programming Interface (API), which supports automatic collection of large data sets.

For our experiments, we chose Twitter profiles as the starting point of our data collection. In Twitter's terminology, the followers of a given user U are users interested in reading U's tweets. These users will be notified when U posts a new tweet. Also, the friends of a user V are the users following V's tweets. In general, users can register themselves as followers of any other user; no permission is required unless the user protects his/her profile using Twitter's protection features. A new Twitter user must first fill a profile form, consisting of about 30 fields containing biographical and other personal information, such as personal interests and hobbies. However, many fields in the form are optional, and indeed substantial portions of Twitter users

leave many or all of those optional fields blank. In addition, Twitter's profile form does not include a specific "gender" field, which complicates gender identification for Twitter users. One can choose additional fields that are not mentioned above for gender classification such as posted tweets; however, we decided to perform gender classification using only profile colors.

Among many other fields in a Twitter profile, here we are interested in the five fields that allow users to choose different colors for the following items of their profiles:

1. Background color.
2. Text color.
3. Link color.
4. Sidebar fill color.
5. Sidebar border color.

Users choose their own preferences by selecting colors from a color wheel while editing their profiles. Unlike other OSNs, such as Facebook, Twitter allows users to redesign and change their profiles. In some cases, users chose both a background color and a background picture (from a picture file) for their profiles. In these cases, the background picture overrides the background color, which is not shown. However, our empirical setup will take into account the background color chosen by a user even if that color is overriden by that user.

We ran our crawler between August and December 2013, subject to Twitter's limitation of less than 150 requests per hour. We started our crawler with a set of random profiles and we continuously added any profile that the crawler encountered (e.g., profiles of users whose names were mentioned in tweets we harvested). Subsequently, we filtered all the profiles with valid URLs. The URL is a profile field that lets a Twitter user create a link to a profile hosted by another OSN, such as Facebook. This field is important because profiles hosted by other OSNs often contain an explicit gender field, which Twitter profiles do not include.

In all, the dataset we used at the time of our study consisted of 169,449 profiles, of which 94,251 were classified as male and 75,198 were classified as female. We considered only profiles for which we obtained gender information independently of Twitter content (i.e., by following links to other profiles). For each profile in the dataset, we collected the five profile colors listed above. We also stratified the data by randomly sampling 150,000 profiles, of which about 75,000 are classified as male and about 75,000 are classified as female. In this manner, we obtain an even baseline containing 50 % male and female profiles. Twitter offers 19 predefined designs, including a default design, to each new user joining the social network. Each design defines colors for all five fields. Users can select those designs easily. As of this writing, the color (R = 192, G = 222, B = 237), a light shade of blue, is the default background color for any new Twitter user.

In order to account for the existence of predefined designs in the Twitter user setup, we have considered different subsets of our overall dataset, and we studied each subset independently of other subsets. In addition, we stratified each subset by randomly sampling the profiles, from which we obtain even baselines

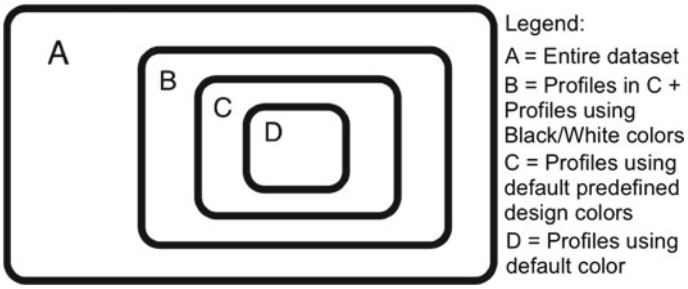


Fig. 1 Four subset of our dataset

containing 50 % male and female profiles. We specifically considered the following subsets:

T1. This is the entire dataset, *A*, consisting of 150,000 profiles with a 50 % male and 50 % female breakdown.
T2. This is dataset *A*–*D*, which is the subset containing all collected profiles, except for profiles using the default design with the RGB values of (192, 222, 237) as the background color, denoted by *D*. *D* represents 11.4 % of dataset *A* while *T2* represents 88.6 %. The base condition for *T2* is a 50 % male and 50 % female breakdown.
T3. This is dataset is *A*–*C*, which is the subset obtained by excluding *C*, the subset all profiles that use any of the 19 predefined designs including the default design, from *A*. *C* represents around 57 % of *A* while *T3* represents 43 %. The base condition for *T3* is a 50 % male and 50 % female breakdown. Here we report detailed empirical results about *T3*, since it includes only profiles with custom color choices, and we summarize results for the other datasets.
T4. This is dataset *A*–*B*, obtained by excluding from the entire dataset, *A*, all profiles, *B*, that use any of the 19 predefined designs as well as black or white as background color. *B* represents 71.8 % of *A*, while *T4* represents 28.2 %. The base condition is still a 50 % male and 50 % female breakdown.

Figure 1 shows the four subsets that we considered for our analyses. Overall, female users are more likely to choose their own layout colors, while male users are more likely to use the default design or one of the other predefined designs.

4 Proposed Approach

Our algorithm for preprocessing colors before feeding the colors to the classifier is shown in Fig. 2. First, we harvest colors from user profiles. Next, we apply a color quantization and sorting procedure (i.e., normalization) to reduce the number of colors. The colors are converted from their Red, Green and Blue (RGB) representation to the corresponding HSV (Hue, Saturation, Value) representation. We then sort

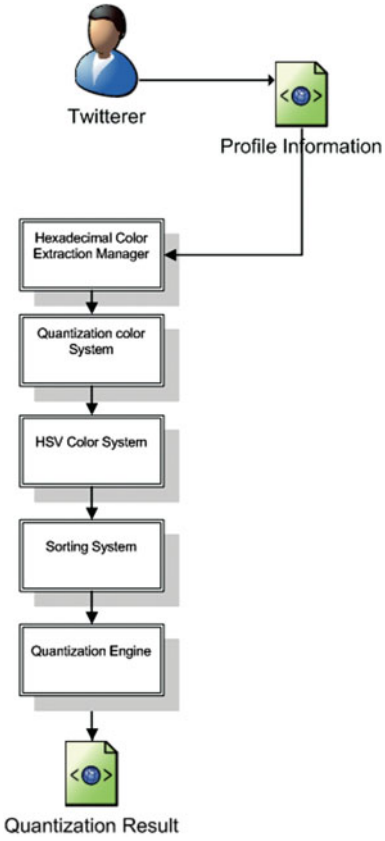


Fig. 2 Algorithm for color preprocessing

the colors by their hue and value, and finally we convert them back to RGB. The sorting allows labeling similar colors (e.g., adjacent colors in the sort) by consecutive numbers that we feed to the classifier.

Figure 3 shows the color distribution of profile background colors harvested from profiles in our data set before quantization. Broader stripes denote the relative

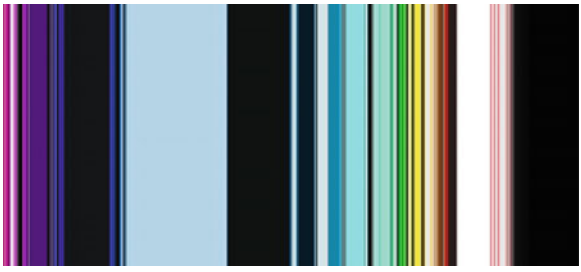


Fig. 3 Distribution of profile background colors before applying color quantization in our dataset

frequency of background color in the profiles that we analyzed. In particular, the broad light blue stripe to the center left of the figure represents the default background color of Twitter profiles.

Colors harvested from Twitter user profiles are typically specified as a combination of RGB values ranging between 0 and 255. This gives a total of 256^3 colors combinations. Because of the large number of combinations, we use quantization, a compression procedure that substantially reduces the huge number of colors. Each of the red, green and blue values is shrunk from 8 bits to 4 bits and 3 bits respectively. This technique reduces the total number of color combinations from $256^3 \approx 16 * 10^6$ to just $16^3 = 4{,}096$ colors and $8^3 = 512$ colors, respectively. Each of the original colors we harvested is converted to the compressed color having the least Euclidean distance from the original color in terms of R,G, and B coordinates. Next, according to the algorithm in Fig. 2, we convert each quantized color to its corresponding HSV representation. We use this representation for sorting the colors according to their similarity. First, colors are sorted by their hue; we use saturation values to break ties between colors having identical hues. Figure 4 shows the 512 colors (i.e., the quantization color procedure of 9-bit RGB) Read, Green, Blue (RGB) obtained after quantization and sorting.

The rationale for applying color quantization is that the feature set obtained from straight RGB values would be quite large, a total of $256^{(3*5)}$ cases for 5 color features. A feature set of this size would be mostly unnecessary as most colors are perceptually indistiguishable from neighboring colors with R, G, and B values differing only by few units from the original color. Thus, we chose to cluster colors in such a way that colors within a given cluster are perceptually similar to each other. Next, we investigated the size of each cluster. Larger clusters would lead to smaller features sets; however, larger clusters may also lead to the inclusion of substantially different colors in the same cluster. For this reason, we studied empirically clusters of various sizes and we concluded that clusters grouping 15^2 colors in each cluster, with 3-bit RGB values per cluster, gave us the highest accuracy results.

We observed empirically that quantization and sorting are beneficial to the accuracy of our gender predictions. In general, our accuracy has improved by up to 15 %

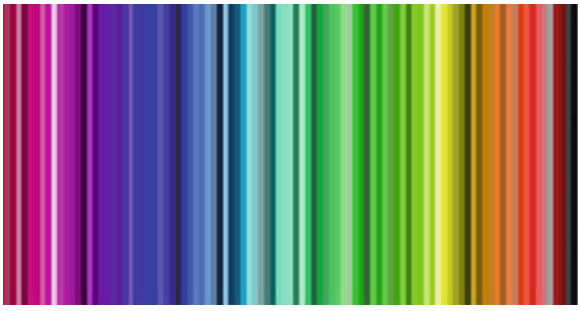


Fig. 4 Spectrum of sorted, quantized colors obtained by the color-preprocessing algorithm shown in Fig. 2

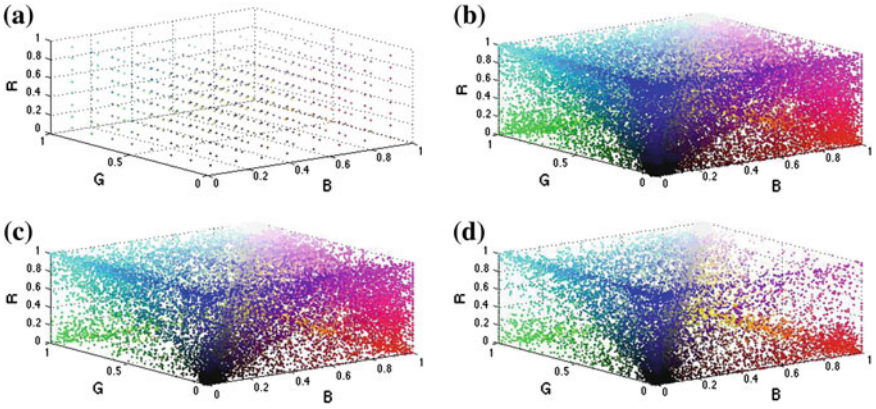


Fig. 5 Part **a** shows the centroid of the quantization color procedure; **b** shows the color distribution of both genders for the profile background after applying the quantization color procedure to our data set; **c** shows the color distribution of the profile background of female users after applying the quantization color procedure to our data set; and **d** shows a similar color distribution for male users

because of these procedures. Figure 5 shows in 3 dimensions the profile background colors distribution for male and female users, the quantization color centroid and background color distribution for both genders in our data set after applying the quantization color procedure of 9-bit RGB. In brief, our quantization color procedure is a reduction from 24-bit to both 12-bit and 9-bit RGB color representations. We tried both finer and coarser representations for colors and we found that 3 bits per color give us the best prediction accuracy among the options that we considered. We conclude that this representation is a reasonable compromise between the number of colors (i.e., the feature values) that we must consider and the perceptual differences within the resulting color clusters. Color quantization is especially important because we are using a total of 5 color features for each user we analyze. In general, quantization reduces the number of cases (i.e., combinations) for five color-based features from $256^{(3*5)}$ cases to $8^{(3*5)}$ cases.

5 Empirical Studies

In this section we evaluate empirically our dataset using different classifiers and we report our findings.

5.1 Experimental Results

We performed four sets of experiments, one for each of the four subsets of our dataset. In each experiment set, we tried many classifiers; different classifiers

produced different results. Next, we selected the top classifiers. Here we consider the following four different classifiers: Probabilistic Neural Network (PNN), Decision Tree (DT), Naïve Bayes (NB) and Naïve Bayes/Decision-Tree Hybrid (NB-Tree). We performed a 10-fold cross validation on our data subsets for each classifier. In each set of experiments, we trained our classifiers with all five color-based features.

We assessed the effectiveness of color quantization by running experiments with and without color quantization (i.e., using the raw RGB data harvested from the Twitter profiles). Tables 1 and 2 report the performance of dataset T3 using different classifiers and color-based features with a 50 % baseline. In particular, we choose T3 among the other datasets because T3 is our largest data subset containing only colors chosen by users from the color wheel. The last five columns in the table report results for different numbers of color features. We use the color features in the order that we listed previously. Thus, the column with one color feature reports only data obtained with the background color alone; the column with two color features reports data for the background color and text color; the next column adds the link color; and the last two columns add sidebar fill and border colors. For each experiment, we report the percentage of correctly identified male users and female users and the overall accuracy.

On the one hand, Table 1 reports the accuracy of gender prediction without applying the quantization and sorting algorithms discussed above. On the other hand, the data in Table 2 was obtained after applying quantization to Twitter profile colors and sorting the resulting color clusters. As shown in Table 1 without quantization, the performance of three color-based features roughly equals the case of four and five features. In the case of the PNN classifier, three features actually give better

Table 1 Accuracy of gender predictions for dataset T3 with RGB colors without quantization

	Scores (%)	1 color	2 colors	3 colors	4 colors	5 colors
NB	Precision	59.2	59.1	61.1	62.1	62.2
	Recall	59.2	59.1	61.1	62.1	62.2
	F-score	59.2	59.1	61.1	62.1	62.2
	Accuracy	59.2	59.1	61.1	62.1	62.2
DT	Precision	59.9	61.5	63.7	64.0	64.1
	Recall	58.8	61.5	63.8	64.0	64.1
	F-score	57.9	61.4	63.8	64.0	64.1
	Accuracy	58.8	61.5	63.8	64.0	64.1
PNN	Precision	**62.2**	**65.6**	**66.7**	66.2	**66.9**
	Recall	**61.2**	**65.7**	**66.5**	65.4	65.0
	F-score	**60.5**	**65.7**	**66.4**	63.2	63.9
	Accuracy	**61.3**	**65.7**	**66.6**	64.4	65.0
NB-Tree	Precision	58.6	61.1	64.4	**67.2**	65.2
	Recall	58.3	61.1	64.4	**67.2**	**65.2**
	F-score	57.9	61.1	64.4	**67.1**	**65.2**
	Accuracy	58.2	61.1	64.4	**67.2**	**65.2**

Table 2 Accuracy of the experiment results for dataset T3 after applying color quantization and sorting

	Scores (%)	1 color	2 colors	3 colors	4 colors	5 colors
NB	Precision	59.1	59.0	61.1	61.9	61.9
	Recall	59.1	59.0	61.1	61.9	61.9
	F-score	59.1	59.0	60.9	61.9	61.9
	Accuracy	59.1	59.0	61.1	61.9	61.9
DT	Precision	61.6	67.4	69.1	68.9	68.5
	Recall	61.3	65.7	68.8	68.7	68.3
	F-score	61.2	64.9	68.6	68.6	68.2
	Accuracy	61.3	65.7	68.8	68.7	68.1
PNN	Precision	61.3	66.2	69.1	68.0	66.6
	Recall	61.2	65.4	69.1	66.8	65.5
	F-score	61.1	65.0	69.1	66.2	65.8
	Accuracy	61.1	65.4	69.1	66.8	66.5
NB-Tree	Precision	**68.7**	**69.8**	**72.7**	**72.5**	**73.9**
	Recall	**69.7**	**68.6**	**72.8**	**72.9**	**73.8**
	F-score	**68.7**	**69.9**	**72.9**	**72.5**	**73.9**
	Accuracy	**70.7**	**71.2**	**73.3**	**73.8**	**74.2**

accuracy than four and five features. Also, in the case of the NB-Tree classifier, four features actually give better accuracy than three and five features. In the case of the NB-Tree classifier, four features provide the best accuracy for the RGB Colors. In contrast with Table 1, in Table 2 the accuracy performance increases when using all five color-based features compared to the cases of three and four color-based features.

On the whole, the data in Tables 1 and 2 show that quantization and sorting of colors result in a significant increase in accuracy, especially when all five-color features are used with Naïve Bayes/Decision-Tree Hybrid (NB-Tree) classifiers and when three-color features are used with the Probabilistic Neural Network (PNN). In fact, these two classifiers obtain overall accuracy results of 74.2 and 69.1 % with quantization and sorting. Without quantization and sorting these two classifiers achieve only 65.2 and 66.6 % accuracy. Modest performance gains are obtained also with the Decision Tree (DT) classifier. In contrast with the other three classifiers, the Naïve Bayes (NB) classifier fails to achieve any gains except the case of the three-color features where it roughly ties its previous performance. In fact, the performance of this classifier drops overall with color quantization and sorting.

Figure 6 shows the accuracy increase obtained by using the color quantization procedure compared to the case of raw RGB colors for each of the four classifiers on dataset *T3*. Part (a) shows the performance of the Naïve Bayes classifier with and without quantization. This is the only classifier that provides slightly better accuracy without quantization than in the case of quantization. However, the overall performance of the classifier is inferior to that of the other classifiers. Part (b) in Fig. 6 shows the performance of the Decision Tree classifier, which yields better accuracy

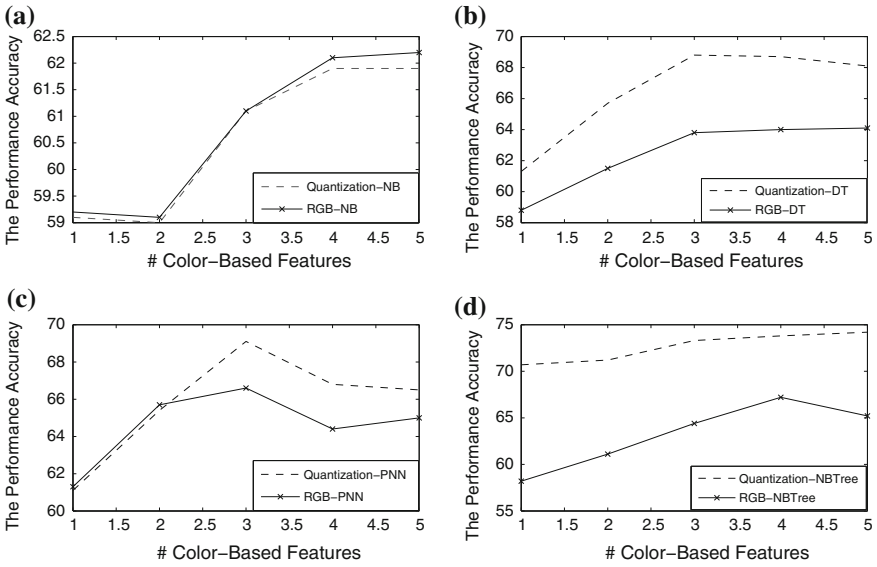


Fig. 6 Accuracy of the four classifiers on dataset T3 using different numbers of color-based features. **a** NB. **b** DT. **c** PNN. **d** NBTree

than Naïve Bayes. In this case, color quantization and sorting improve slightly the accuracy of the predictions. The performance of the Probabilistic Neural Network (PNN) Probabilistic Neural Network and Naïve Bayes/Decision-Tree Hybrid (NB-Tree) classifiers are shown in Part (c) and Part (d) of Fig. 6.

Table 3 shows the performance of the four classifiers on all four datasets that we considered after color quantization. Evidently, NB-Tree has the best accuracy on all five datasets with accuracy results consistently above 70 % in all four cases. We specifically obtained our best results with the NB-tree classifier in the *T3* dataset with an accuracy of 74.2 % over a 50 % baseline of both genders, a gain of about 24.2 %.

An advantage of our approach is that it uses only five colors, making it language independent. An additional advantage is that it has a low-dimensional space, resulting in a low computational complexity of our classifiers. In contrast with our method, most existing approaches are language dependent while using high dimensional spaces generated from unique words extracted from text (i.e., tweets, names, and profile descriptions), and millions of features. For instance, Burger et al. [3] utilize 15.6 million features with each feature corresponding to a unique word extracted from a tweet. Similarly, Rao et al. [6] use 1.25 million features extracted from tweets.

Figure 7 shows the difference in colors chosen by female versus male Twitter users. On the top we show popular colors chosen by female users (after clustering); the colors for male users are shown on the bottom of the figure.

Figure 8 shows the effects of different training set sizes on the accuracy of the predictions. Similar to Fig. 6, the four parts of the figure refer to different classifiers; for each classifier we use color-coded lines to distinguish the number of color features that we consider.

Table 3 Accuracy of the experimental results for the four different datasets with color quantization and sorting

	Scores (%)	T1	T2	T3	T4
NB	Precision	64.1	63.0	61.9	62.7
	Recall	64.2	63.1	61.9	62.7
	F-score	64.2	63.1	61.9	62.7
	Accuracy	64.3	63.2	61.9	62.6
DT	Precision	69.30	69.3	68.5	61.4
	Recall	68.9	69.5	68.3	60
	F-score	69.9	69.4	68.2	60.7
	Accuracy	69.9	69.5	68.1	63.8
PNN	Precision	62.0	67.6	66.6	67.3
	Recall	61.4	65.6	63.5	64.6
	F-score	61.0	64.6	61.8	63.2
	Accuracy	61.4	65.6	63.5	64.6
NB-Tree	Precision	**72.3**	**71.6**	**73.9**	**71.9**
	Recall	**72.0**	**71.4**	**73.8**	**71.4**
	F-score	**72.1**	**71.5**	**73.9**	**71.2**
	Accuracy	**72.3**	**72.0**	**74.2**	**71.4**

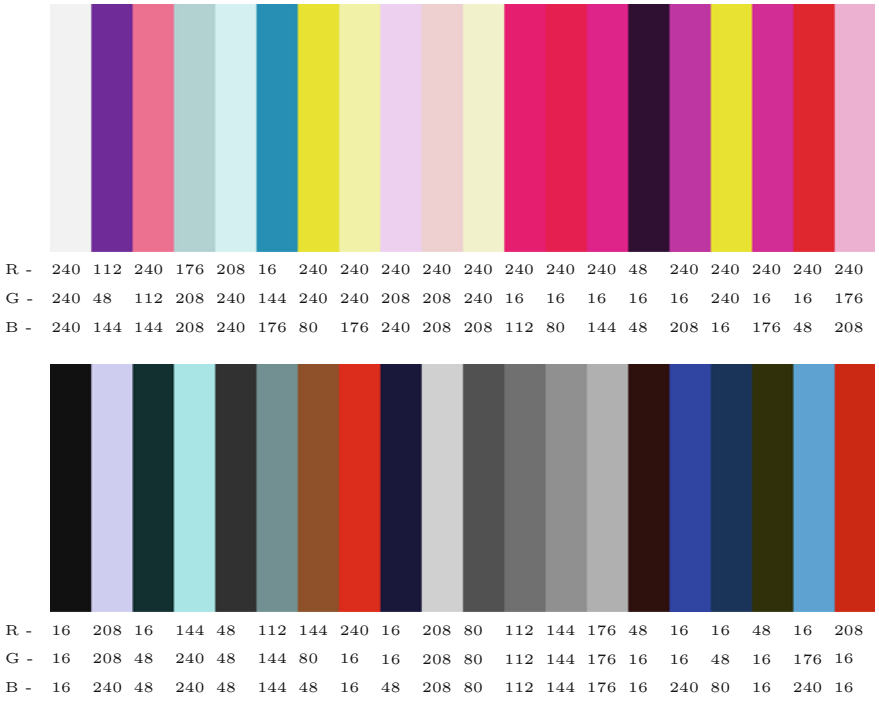


Fig. 7 Spectrum of popular colors for female users (*top*) and male users (*bottom*)

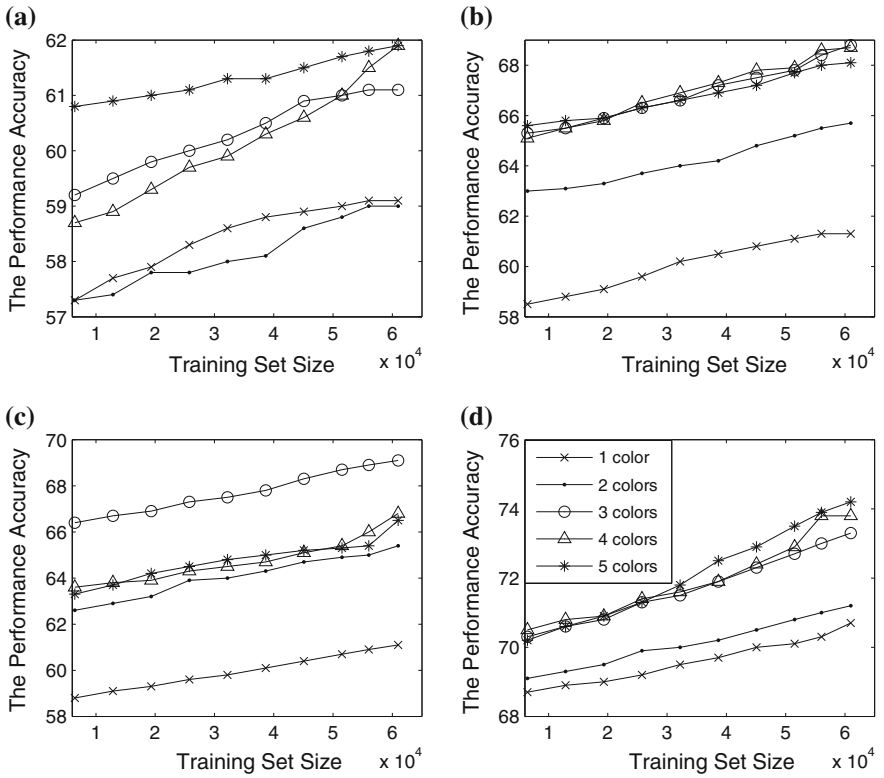


Fig. 8 Effects of different training set sizes on accuracy of different classifiers on dataset T3 with different numbers of color-based features. **a** NB. **b** DT. **c** PNN. **d** NBTree

All diagrams refer to data set T3. In general, the accuracy of our predictions grows linearly in the size of the training sets; larger training sets yield better accuracy results. The four classifiers exhibit similar behaviors with respect to training set size. However, the performance of the classifiers differs depending on the number colors considered. In particular, the PNN classifier does best when three colors are used. Evidently, the inclusion of the sidebar fill color and border color has an adverse effect on the performance of this classifier. The DT and NB-Tree classifiers exhibit similar performance in the case of three, four and five colors. The performance of the DT classifier drops significantly when two colors are used, even more so in the case of one color. The NB-Tree classifier also exhibits a performance drop in the case of two colors and one color; however, this classifier appears to be less sensitive to the number of colors than the DT classifier. Finally, the NB classifier shows the worst performance of the four classifiers we considered; however, this classifier benefits when larger color sets (consisting of 5 and 4 colors) are used. We conclude that the NB-Tree classifier is the most suitable for our gender predictions. Not only does this classifier yield the highest accuracy results; it is also more robust than the other classifiers when fewer colors are considered.

5.2 Threats to Validity

There are two main threats to the validity of this study. The first threat is our reliance on self-declared gender information entered by Twitter users on external web sites for validation of our predictions. We use this gender information as our ground truth. Evidently, a complete evaluation of all 169,449 Twitter users would be impractical. We manually spot-checked about 10,000 out of the 169,449 profiles in our dataset or about 6.0 % of the dataset. In the cases that we checked by hand, we are confident that the gender information we harvested automatically was indeed correct about 95 % of the time. The second threat is given by the overall size of the dataset that we could analyze. Although we started from four million Twitter users, we ended up with just 169,449 users whose gender we could verify independently. This indicates that the size of the training sets was adequate; however, we will continue expanding our data set. Apparently, little will be gained by using larger datasets.

6 Conclusions and Future Work

In this paper, we studied gender classification on Twitter. We presented a novel approach for predicting gender utilizing only five color-based features extracted from profile layout colors. Unlike existing works that use millions of features, we used only five color-based features. Despite the challenging feature-based characteristics for gender classification, we proposed a color-based model for gender classification. We applied quantization colors procedure to the color-based features that compressed the color from 24-bits to 9-bits and produced a discrete set of 512 colors. We empirically proved the validity of our approach by examining different classifiers over a large Twitter data set. Our approach is using an agent with advanced colors preferences to search all profiles and predicting gender. Our empirical studies show that our method is reasonably accurate and highly efficient in terms of computational complexity.

In the future, we intend to study different characteristics of the dataset to classify gender (e.g., features of a user's friends and followers) and to incorporate them with the profile's colors.

References

1. Mocanu D, Baronchelli A, Perra N, Gonçalves B, Zhang Q, Vespignani A (2013) The Twitter of Babel: mapping world languages through microblogging platforms. PLoS One 8(4):e61981
2. Wauters R, Only 50% of Twitter messages are in English, study says. http://techcrunch.com/2010/02/24/twitter-languages/
3. Burger JD, Henderson J, Kim G, Zarrella G (2011) Discriminating gender on Twitter. In: Proceedings of the 2011 conference on empirical methods in natural language processing. Edinburgh, Scotland, UK. Association for Computational Linguistics, July 2011, pp 1301–1309. [Online] http://www.aclweb.org/anthology/D11-1120

4. Al Zamal F, Liu W, Ruths D (2012) Homophily and latent attribute inference: Inferring latent attributes of Twitter users from neighbors. In: 6th international AAAI conference on weblogs and social media (ICWSM'12), 2012
5. Liu W, Al Zamal F, Ruths D (2012) Using social media to infer gender composition of commuter populations. In: Proceedings of the when the city meets the citizen workshop, the international conference on weblogs and social media
6. Rao D, Yarowsky D, Shreevats A, Gupta M (2010) Classifying latent user attributes in Twitter. In: Proceedings of the 2nd international workshop on search and mining user-generated contents, pp 37–44
7. Liu W, Ruths D (2013) What's in a name? Using first names as features for gender inference in Twitter. In: 2013 AAAI spring symposium series, in symposium on analyzing microtext
8. Alowibdi J, Buy U, Yu P (2013) Empirical evaluation of profile characteristics gender classification on Twitter. In: The 12th international conference on machine learning and applications (ICMLA), vol 1, pp 365–369, December 2013
9. Alowibdi J, Buy U, Yu P (2013) Language independent gender classification on Twitter. In: IEEE/ACM international conference on advances in social networks analysis and mining, ASONAM'13, pp 739–743, August 2013
10. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The weka data mining software: an update. ACM SIGKDD Explor Newsl 11(1):10–18
11. Berthold MR, Cebron N, Dill F, Gabriel TR, Kötter T, Meinl T, Ohl P, Thiel K, Wiswedel B (2009) Knime-the konstanz information miner: version 2.0 and beyond. ACM SIGKDD Explor Newsl 11(1):26–31
12. Singh S (2001) A pilot study on gender differences in conversational speech on lexical richness measures. Lit Linguist Comput 16(3):251–264
13. Argamon S, Koppel M, Fine J, Shimoni AR (2003) Gender, genre, and writing style in formal written texts. Text 23(3):321–346
14. Koppel M, Argamon S, Shimoni AR (2002) Automatically categorizing written texts by author gender. Lit Linguist Comput 17(4):401–412
15. Sarawgi R, Gajulapalli K, Choi Y (2011) Gender attribution: tracing stylometric evidence beyond topic and genre. In: Proceedings of the fifteenth conference on computational natural language learning, Portland, OR, pp 78–86, June 2011
16. Nowson S, Oberlander J, Gill A (2005) Weblogs, genres and individual differences. In: Proceedings of the 27th annual meeting of the cognitive science society, Stresa, Italy, pp 1666–1671
17. Kucukyilmaz T, Cambazoglu BB, Aykanat C, Can F (2006) Chat mining for gender prediction. Advances in information systems. Springer, Berlin, pp 274–283
18. Mukherjee A, Liu B (2010) Improving gender classification of blog authors. In: Proceedings of the 2010 conference on empirical methods in natural language, processing. Association for Computational Linguistics, Cambridge, MA, pp 207–217, October 2010. [online]. http://www.aclweb.org/anthology/D10-1021
19. Peersman C, Daelemans W, Van Vaerenbergh L (2011) Predicting age and gender in online social networks. In: Proceedings of the 3rd international workshop on search and mining user-generated contents, pp 37–44
20. Herring SC, Paolillo JC (2006) Gender and genre variation in weblogs. J Socioling 10(4):439–459
21. Brain S, Twitter statistics. http://www.statisticbrain.com/twitter-statistics
22. Business T, Who is on Twitter? https://business.twitter.com/whos-twitter

TUCAN: Twitter User Centric ANalyzer

Luigi Grimaudo, Han Hee Song, Mario Baldi, Marco Mellia and Maurizio Munafò

Abstract Twitter has attracted millions of users that generate a humongous flow of information at constant pace. The research community has thus started proposing tools to extract meaningful information from tweets. In this paper, we take a different angle from the mainstream of previous work: we explicitly target the analysis of the timeline of tweets from "single users". We define a framework—named TUCAN—to compare information offered by the target users over time, and to pinpoint recurrent topics or topics of interest. First, tweets belonging to the same time window are aggregated into "bird songs". Several filtering procedures can be selected to remove stop-words and reduce noise. Then, each pair of bird songs is compared using a similarity score to automatically highlight the most common terms, thus highlighting recurrent or persistent topics. TUCAN can be naturally applied to compare bird song pairs generated from timelines of different users. By showing actual results for both public profiles and anonymous users, we show how TUCAN is useful to highlight meaningful information from a target user's Twitter timeline.

Keywords Microblog · Topic models · Topic analysis · Web mining

L. Grimaudo (✉) · M. Mellia · M. Munafò
Politecnico di Torino, Torino, Italy
e-mail: luigi.grimaudo@polito.it

M. Mellia
e-mail: mellia@polito.it

M. Munafò
e-mail: munafo@polito.it

H.H. Song · M. Baldi
Narus Inc., Sunnyvale, CA, USA
e-mail: hsong@narus.com

M. Baldi
e-mail: mbaldi@narus.com

J. Kawash (ed.), *Online Social Media Analysis and Visualization*,
Lecture Notes in Social Networks, DOI 10.1007/978-3-319-13590-8_4

1 Introduction and Motivation

Twitter is nowadays part of everyone's life, with hundreds of millions of people using it on regular basis. Originally born as a microblogging service, Twitter is now being used to chat, to discuss, to run polls, to collect feedback, etc. It is not surprising then that the interest of the research community has been attracted to study the "social aspects" of Twitter. User and usage characterization [1, 2], topic analysis [3–5], community-level social interest identification [1] have recently emerged as hot research topics. Most of previous works focus on the analysis of "a community of twitters", whose tweets are analysed using text and data mining techniques to identify the topics, moods, or interests.

In this paper we take a different angle: first, we focus on the analysis of a Twitter *target user*. We consider set of tweets that appear on his Twitter public page, i.e., the target user's timeline, and define a methodology to explore exposed content and extract possible valuable information. Which are the tweets that carry the most valuable information? Which are the topics he/she is interested into? How do this topics change over time? Our second goal is to compare the Twitter activity of two (or more) target users. Do they share some common traits? Is there any shared interest? What is the most common interest of these two users, regardless of the time they are interested in it?

We propose a graphical framework which we term as TUCAN-Twitter User Centric ANalyzer. TUCAN highlights correlations among tweets using intuitive visualization, allowing exploration of the information exposed in them, thus enabling the extraction of valuable information from user's timeline. Given a number of limitations on the topic analysis of Twitter messages, such as limited length of messages, prevalent use of non-dictionary words (i.e., abbreviations, mentions, hashtags, retweets, slang, and cultural words), and lack of contextual resources (e.g., due to extensive use of Twitter for "private" purposes [6]), lots of ingenuity is required to automatically extract significant information out of tweets. From a methodology stand-point, we build upon text mining techniques, adapting them to cope with the specific Twitter characteristics.

As input, we group a user's tweets based on a window of time (e.g., a day, or a week) so to form *bird songs*, one for each time window. At the next step, filtering is applied to each bird song using either simple stop-word removal, stemming, lemmatization, or more complicated transformations based on lexical databases. Next, terms in bird songs are scored using classic Term Frequency-Inverse Document Frequency (TF-IDF) [7] to pinpoint those terms that are particularly important for the target user. Each pair of birds songs are finally compared by computing a similarity score, so to unveil those bird songs that contain overlapping, and thus persistent, topics. The output is then represented using a coloured matrix, in which cell colour represents the similarity score. As a result, TUCAN offers a simple and natural visual representation of extracted information that easily unveils the most interesting bird songs and the persistent topics the target user is interested into during a given time period. Moreover, comparisons among bird songs gives intuitions on the transition of user interests as well as the significance of topics to the user.

The framework is naturally extended to find and extract similarities among tweets of two or more target users. TUCAN computes and graphically shows the similarity among bird songs generated from the timelines of the pairs of target users, revealing similarities and common interests that are present possibly during different time periods.

TUCAN demonstrates to be useful to highlight correlation among tweets, which in turn proves very valuable in identifying topics of interest in the Twitter timeline of a user. This is very instrumental in generic individual profiling or surveillance applications, where the information hidden inside the target user's flow of tweets has to naturally emerge. TUCAN is also very powerful to compare individuals, to examine their timelines in parallel, hunting for similarities, pinpointing common interests, and observing changes, deviations, etc. For instance, comparing a well-known public profile timeline, e.g., President Barack Obama, against a generic target user would unveil if they share common political interests. Alternatively, two casual targets can be compared to see if some common trait/interest exist (possibly at different time), e.g., to evaluate the success of an Internet dating or marriage.

To demonstrate the effectiveness of TUCAN on real-world microblogs, we applied it to two month long history of 712 Twitter users. Results show that the correlation among tweets turns out to be a key point in the identification and analysis of twitter users over time; analyzing tweet messages of a politician, we were able to confirm that his topics and topic durations well matched with ongoing political events at the time. Comparing his tweets against tweets from the US government, a subset of topics that are in-line with the government's positions were picked up. Analysis on topic changes revealed transitions in users' social relationships.

Part of this paper [8], it was presented as a poster in *The 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*. This version improves the performance evaluation of TUCAN by presenting a parameter sensitivity analysis, a discussion of recent related works and a more detailed set of results.

2 Related Work

The increasing availability of valuable information from microblogging platforms pushed the research community to investigate efforts for mining textual information from them.

Text topic extraction and modeling. A plurality of works [6, 9–13] is based on a well known topic modeling technique called, the Latent Dirichlet Allocation (LDA) [14]. Chang et al. [10] extends LDA to infer descriptions of entities (e.g., authors) separately from their relationships. Ramage et al. [6] incorporates supervision to LDA, leveraging hashtags of Twitter for topic labeling. Generalizing topic extraction to Tweets without hashtags, [11] directly applies LDA to individual sentence within each Tweet message.

To further enhance the performance of topic extraction from short and sparse messages, author-topic (AT) model was proposed [15, 16]. By creating topic mixture at the level of authors rather than individual documents, AT is claimed to obtain more stable set of topics than LDA. Hong and Davison [4] conducts empirical comparisons of LDA, AT, and simple TF-IDF on aggregates of Tweet messages. The work discovers that the accuracy of the topic models are highly influenced by the length of the documents. It also finds that with long enough documents, the model based approaches become less effective compared to the baseline TF-IDF. Based on the observations, we design TUCAN to flexibly aggregate messages into bird songs. With effectively formed bird songs, TUCAN can provide powerful topic analysis even with generic TF-IDF.

Time-series analysis in microblogs. Many literatures on topic analysis [3, 6, 11] focus on detecting emergence of anomalous topics or prominent shifts on topic trends. Leveraging groups of semantically associated document tags, [3] discovers temporally emergent topics from Twitter data stream. Ramage et al. [6] defines four types of Tweet categories and classifies streamed messages into them. Because these time-series analysis work on the entire group of users as a whole and do not distinguish single users, they cannot express topical relationships across individuals. We, on the other hand, focus on building dynamic relationships among the users. Aimed at similar goal, [17] proposes to detect topical relationships across entities over time. However, they only focus on time correlated co-occurring events. Instead, TUCAN aims to detect topic correlations even if they occur at different time frames. Recently, [18] proposed an interactive tool to analyse topic extracted from a stream of tweets organized in adjacent time slices of equal length. LDA is applied to mine topics and cosine similarity is leveraged to align them from the different time bins. Again, TUCAN is more user-centric and flexible enough to reveal topic correlations from time periods not strictly consecutive in time.

3 Framework

The TUCAN architecture includes three modules: (i) bird song generator, (ii) cross-correlation computation engine, and (iii) dashboard visualizer. A set of target Twitter users, e.g., their screen names or user-ids, is provided to the system as an input. The system collects tweets related to such users on which various analytics are executed. Their outcome is visualized to enable the operator to gain knowledge about the users and the topics they are twitting about.

3.1 Bird Song Generation and Cleaning Process

Let $TW(u)$ be the set of tweets of a single user u that are retrieved from Twitter, time stamped with their generation time, stored and organized in a repository in binary

format, to be easily accessed and further analyzed when necessary. Bird songs are created by aggregating tweets from $TW(u)$ generated within a time period T, to then be analyzed. We define the ith *bird song* for the user u, $BS(u, i)$, as the subset of tweets in $TW(u)$ that appear in the ith time period of duration T, i.e., the set of tweets that are generated in the $[(i-1)T, (i)T),\ i > 0$ window of time.

A "plain cleaning" pre-processing is applied to bird songs to discard stop-words, HTML tag entities, and links. Plain cleaning can be possibly substituted by more advanced text cleaning mechanisms; the following are also considered in this work: (i) removal of Twitter 'mentions', (ii) stemming, (iii) lemmatization, and (iv) ontology-based lexicon generalization. TUCAN allows the analyst to select the most appropriate cleaning method to take advantage of different effects of them in different contexts. Twitter mentions are words that begins with @ signs representing the mentioning of some named entities. The intuition behind removing the mentions comes from the fact that they do not provide insight in the topics being addressed, being just Twitter-ID of other users. Stemming and lemmatization are common text processing techniques aiming at reducing a word to its root form to lower sparseness present in a text document. The main difference between stemming and lemmatization is that the former is based on the heuristic of removing the trailing part of a word, while the latter brings a word to a canonical form based on a vocabulary and a morphological analysis the word. Here the Porter stemming algorithm [19] was deployed, while lemmatization is derived from the well-established Wordnet lexical database [20]. At last, our ontology-based lexicon generalization method leverages the Wordnet database to derive the most general concept for each word in the bird song. For instance, "gun" and "rifle" are replaced by the more generic term "weapon". The impact of the different cleaning methods will be exemplified by the experimental results presented in Sect. 4.

3.2 Cross-Correlation Computation

Each pre-processed bird song is tailored in a Bag-Of-Words (BoW) model, a common representation used in information retrieval and natural language processing. The bird song is tokenized in an unordered set of words, disregarding their sequence and position. Each word is then scored according to a weighting scheme. In this work, the Term Frequency-Inverse Document Frequency (TF-IDF) score is adopted as past literature has shown it to produce good results [4]. TF-IDF is computed as the product of the frequency of a term in its bird song and the inverse of the frequency of the term in the set of documents (i.e., all bird songs) being analyzed. TF-IDF provides a measure of the importance of a term in a specific bird song (first factor) put in perspective with how common the term is in the whole collection of bird songs. The intuition behind this weighting scheme is that, if a word appears in a huge number of bird songs in a given collection, its discriminative power is very low and is probably not useful to represent the content of the bird song, even if it often appears in it. Hence, words that are frequent in a bird song but rare in the collection are assigned with higher weights.

Bird songs are then transformed into a vector space model $VS(u, i)$, in which each word is given a fixed position. In this space, each word in the bird song $BS(u, i)$ is characterized by its TF-IDF score. Words that do not appear in $BS(u, i)$ are characterized by a null score.

To evaluate the similarity $VS(u, i) \otimes VS(v, j)$ among a pair of bird song vectors the Cosine similarity measure is applied.

Given any two term document vectors, the Cosine similarity is the cosine of the angle between them. The closer two vectors are to one other, the smaller the angle between them will be, i.e., the higher their similarity. Intuitively, the Cosine similarity of two very similar bird songs will be close to one. Instead, if no common words appear in two bird songs, their Cosine similarity will be 0.

3.3 Dashboard Visualizer

In order to pinpoint similarities among bird songs, independently of the time the user posted them, TUCAN computes the similarity score for all possible pairs of bird songs. In total, N^2 similarity scores are computed and stored in a matrix form, where each cell represents $VS(u, i) \otimes VS(u, j), i, j \in [1, N]$. To help identifying correlation, the matrix is presented to the analyst in a graphical format using a web interface. Each cell is represented by a square whose color reflects the similarity score between the ith and jth bird songs. In particular, let

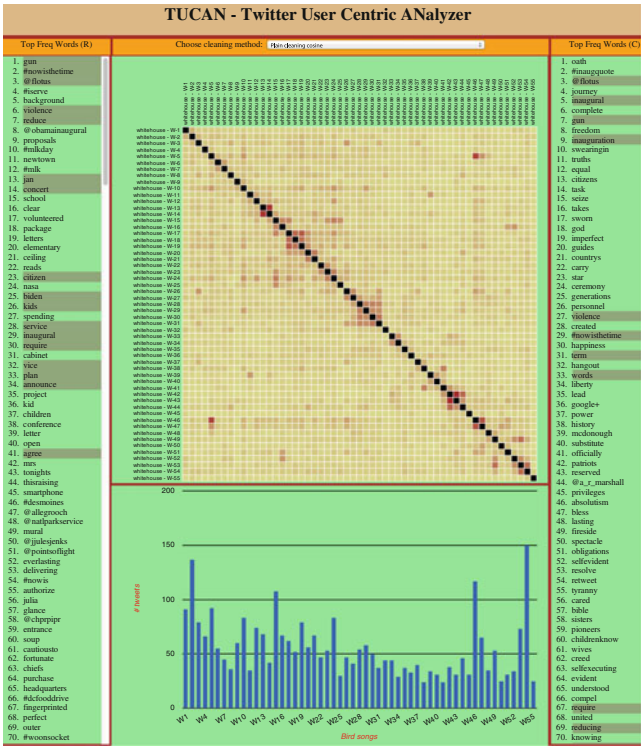
$$m = \max_{i,j,i \neq j} VS(u, i) \otimes VS(u, j),$$

cells are colored with different intensity, using a linear scale, so that the cell with similarity equal to m has the darkest color (see Fig. 1a for an example). Bird songs are organized in increasing time window from left to right (and top to bottom).

As shown in Fig. 1a, when a cell is clicked, the web interface displays the top-ranked words appearing in $BS(u, i)$ and $BS(u, j), i \neq j$ on the left and right panes next to the matrix. Words that appear in both bird songs are highlighted. When clicking on the cells in the main diagonal (presented always in black[1]), the analyst is offered a popup showing the content of the original tweets of the ith bird song. The GUI also shows a histogram below the matrix reporting $n(u, i)\ \forall i$ to allow the analyst to easily gauging variations in the bird song size, e.g., due to the user changing his twitting habits during a holiday period. At the top of the matrix, the analyst is offered a drop-down menu to select the cleaning pre-processing to be applied.

[1] Note that by definition, $VS(u, i) \otimes VS(u, i) = 1$.

(a)



(b)

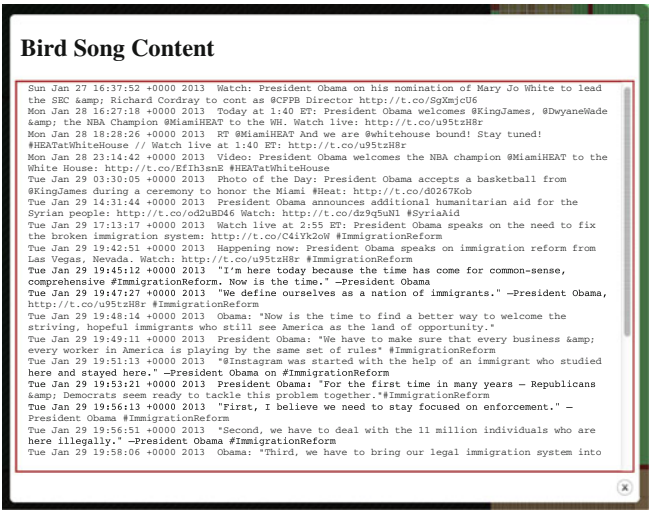


Fig. 1 TUCAN Web Interface showing the analysis of the WhiteHouse official account. $T = 7$ days, plaing cleaning and Cosine similarity are considered. **a** TUCAN Main Interface. **b** Bird song detail

4 Experiments

Applying TUCAN to real world data from Twitter, we conducted an extensive study examining its capability on analyzing user centric topics. We begin by presenting a description on our dataset and how we collected it. Then we provide a series of sensitivity evaluation on various parameters of TUCAN, followed by a number of use cases with emphasis on different aspects of user centric topic analysis.

4.1 Dataset Description

To perform user centric analysis through TUCAN, we monitor 712 randomly selected Twitter users for two or more months starting from the Summer 2012. The actual Tweet period covered for each user depends on the combination of the user's activity and crawling limitations imposed by Twitter API. Additionally, we monitor 28 well-known public figures, selected among politicians, news media, tech blogs, etc. In total, we collect 740 twitter timelines leveraging Twitter REST APIs.[2] Specifically, we access each user's public timeline and retrieve tweet STATUS objects which contains monograms (messages he puts on his page with no destined user), mentions of other users, conversations with follower/followers, and status updates.

From a total of 810,655 tweets, it emerges that 15 % of them contain hashtags, 25 % contain replies and 12 % hyperlinks to other web pages. Similar proportions of message types are reported in the literature, suggesting our dataset presents no bias towards any particular types of tweets. About 300 users (40 %) twitted more than twice in each week. Out of them, 20 users posted more than 400 tweets per week (i.e., more than 57 tweets/day). This already suggests that the window size parameter T has to be tailored to each user twitting habit when forming bird songs. Section 4.3 presents sensitivity tests on T.

4.2 The TUCAN GUI

Revisiting Fig. 1, we present how TUCAN GUI is used for our analysis with an example of 56 week long history of official White House tweets. The reader can appreciate the correlation that TUCAN highlights among bird songs along the main diagonal. The darker areas indeed show that the correlation among top-words in bird songs is high, unveiling persistent topics. For instance, the top-words presented in the left and right lists easily allow to see the topics the White House was twitting about, i.e., violence and inauguration (Week-41). Those tweets refer to the second half of January 2013 during (i) the Inaugural Address by President Barack Obama,

[2] https://dev.twitter.com/docs/api.

and (ii) the debate on violence and weapon possession started after the Newtown school tragedy. For reference, consider (part of) the tweets that form the bird song referring the 21st of January 2013 on Fig. 1b. Intuitively, extracting and summarizing information from the original tweets is much more complicated than by observing TUCAN output. Other areas of high correlation are clearly visible. Those refer to the Sandy hurricane, London Olympics games, etc. TUCAN allows to easily spot these major events that last for several weeks. Notice the Week-6/Week-46 dot with high similarity. Topics in those weeks refer to bills, insurance, gas price, and cost of education.

4.3 Parameter Sensitivity Analysis

We begin our analysis on TUCAN by showing effects of tuning different parameters: time window sizes, preprocessing methods, and inclusion of Twitter mentions. Results are presented showing, for all bird songs pairs of user u, the similarity score sorted in decreasing order. The X-axis displays the bird song rank normalized to the number of bird songs $N(u)$. The Y-axis shows absolute values of similarity score.

Effect of different time window sizes. The time window size T determines the size of bird song—a highly important parameter for topic models to perform optimally [4]. Figure 2 shows comparisons of time windows sizes for a public figure (Barack Obama, on the top) and a randomly chosen normal user (User A, on the bottom). As we vary window size from $T = 1$ day to $T = 14$ days, we expect that the overall similarity scores become strictly higher. Indeed, Fig. 2 clearly shows this; for instance, for Barack Obama, the average (max) score of $T = 1$ day is 0.03 (0.87), average score of $T = 14$ days is 0.11 (0.38). Same observation holds for normal users as shown in Fig. 2b. Notice that higher similarity score is not always welcome; a too large aggregation time window tends to create very large birds songs, in which

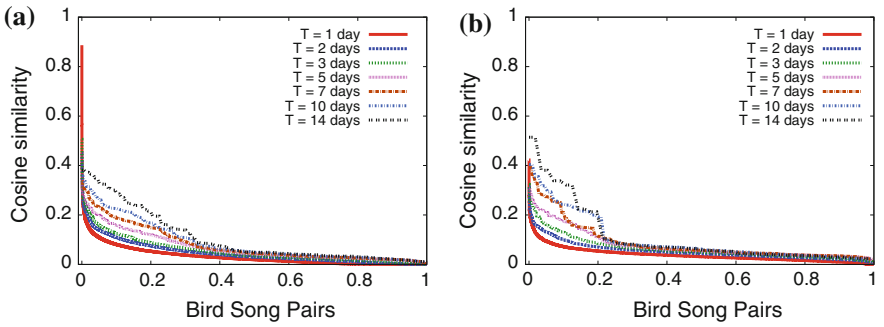


Fig. 2 Effect of different time window sizes T. Plain cleaning and Cosine similarity. **a** Barack Obama. **b** User A

similarity is artificially inflated, and the analysis blurred. As previously stated, T should be matched to twitting habits of the target.

On the other end, too short aggregation time window makes similarity interesting only on a small subset of bird song pairs, focusing the analysis on a too small groups of bird songs. Artifacts are also possibly created. For instance, notice the high similarity score at $x = 0$ in Fig. 2a when $T = 1$ day. The reason for this outlier is that bird songs are formed by only a handful of terms; if three or four of those happened to co-occur in two bird songs, their similarity score turns out to be extremely high.

Further inspection on topic words also supports the importance of aggregation of tweets into bird songs. Table 1 shows up to ten top-words extracted from Barack Obama's bird songs (as previously mentioned). When tweets are used as they are (without aggregation), we not only observe that the number of common words are small, i.e., the tweet has too few words to allow successful analysis; but we also observe that the relationship among the words are loose. Similarly, for $T = 1$ day, no clear topic emerge. In contrary, when $T = 7$ or $T = 14$ days, the top-words are much more coherent (especially between 'gun', 'violence', and 'broken') pinpointing to a clear topic.

In summary, both the general trend of small similarity, and possible existence of outliers suggest to use quite large time window for analysis. As observed in Table 1, and from other test run on large number of users, $T = 7$ days usually gives the same amount of meaningful keywords as larger window sizes (e.g., $T = 14$ days), and gives an overview of weekly pattern of users' activities. For that reason, from here on, we use $T = 7$ days unless otherwise noted. Once similarity has been pinpointed, the analyst can drill down by lowering the value of T.

Effect of different pre-processing methods. Many researchers on information extraction have proposed different pre-processing methods to sanitize original documents. On the particular application to Twitter document analysis, however, no work identified the optimal method. Therefore, we evaluate the performance of three well-known sanitization methods—stemming, lemmatization, lexical generalization—

Table 1 Top-words ranked by TF-IDF, Barack Obama

Rank	Single tweet	$T = 1$ day	$T = 7$ days	$T = 14$ days
1	Photo	Lead	#immigrationreform	#immigrationreform
2	Day	International	**Immigration**	**Gun**
3	Bo	@cfpb	**Gun**	**Immigration**
4	Snow	Cordray	**Violence**	**Violence**
5		Mary	Comprehensive	Comprehensive
6		Snow	@whlive	@whlive
7		Nominates	Broken	Broken
8		Sec	@vp	Reform
9		Richard	Representative	Representative
10		White	Reform	@vp

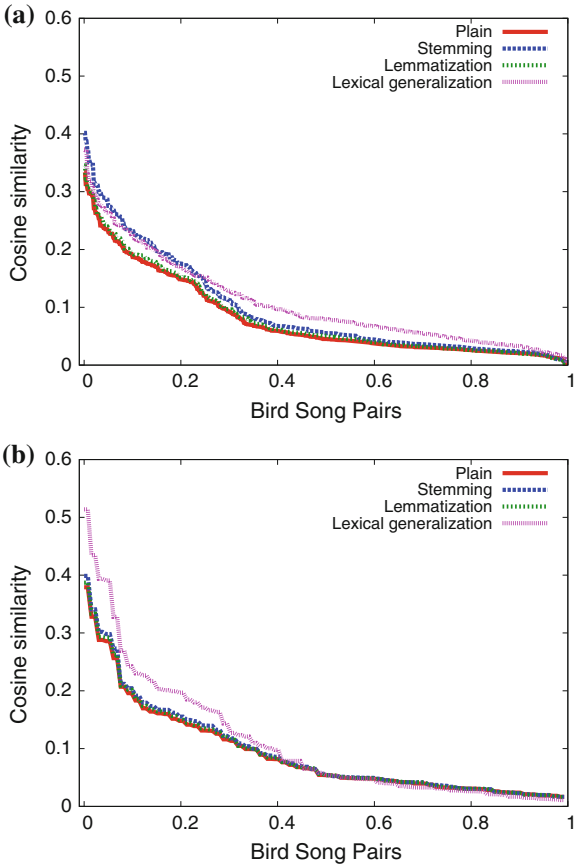


Fig. 3 Effect of different cleaning methods. Cosine similarity and $T = 7$ days. **a** Barack Obama. **b** User B

applied on the top of plain cleaning. Figure 3 compares the cleaning methods considering one public and one normal Twitter users as before. For the profile of Barack Obama, Fig. 3a suggests that stemming and lexical generalization work better than lemmatization and plain cleaning. However, the overall gap between the two groups of curves is less than 0.05 in similarity score. In the case of a normal user, Fig. 3b shows that lexical generalization tend to perform better than other pre-processing methods (by about 0.05). Notice that the increase in similarity is predominant for those pairs whose similarity is already quite large, and thus possibly less useful. By investigating further, we notice that lexical generalization tends (by definition) to return more general topics. In summary, we observe small impact of the filtering process, and results are marginally affected by this choice. As such, TUCAN has been designed to offer the analyst the choice of the cleaning method that he consider the best for the case under analysis. We use plain cleaning as our default choice as the simple preprocessing is sufficient for topic analysis for our dataset.

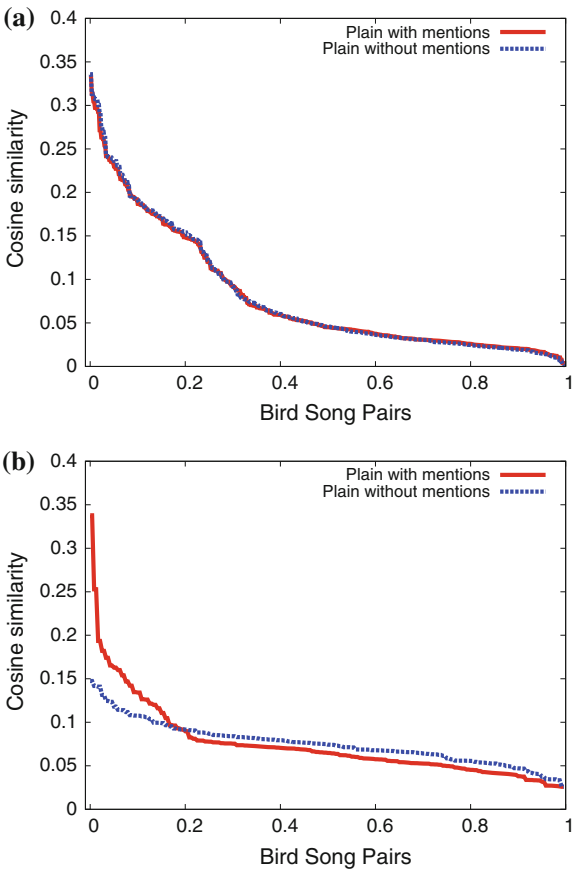


Fig. 4 Effect of mention removal. $T = 7$ days, plain cleaning, and Cosine similarity. **a** Barack Obama. **b** User C

Effect of including Twitter mentions. Among many specific mechanisms Twitter offers, "mentions" play an important role in the analysis of user conversations [21]. From our analysis, we noticed an interesting contrasts when mentions are included or excluded. As Fig. 4a shows, for public figure's tweets (Barack Obama), results of including and not including mentions do not make much difference in similarity distributions. This is because of the usage of mentions by public profiles: either those are rarely used (e.g., in news media), or they are used to mention (i) to lots of different users, or (ii) to always the same group of users (this is the case for Barack Obama). However, for a normal user, as seen in Fig. 4b, proportion of mentions can get up to 70 % and clearly makes distinction on the similarity distribution. The reason for similarity being higher when mentions are included is because the mentions themselves works as keywords (as in the case of '@whlive' or '@vp' from Table 1 that are however the Twitter profiles of White House Live and of the Vice President), resulting in (unnaturally) increased similarity scores. In Sect. 4.4, we will demonstrate cases where inclusion of mentions can indicate a particular pattern of a normal

user's social relationship. Unless explicitly denoted, however, we include mentions in our analysis.

4.4 User Centric Analysis

To demonstrate the effectiveness of TUCAN on user analysis, we present results of case studies. Unless mentioned otherwise, we use the following settings by default: (i) windows size of 7 days, (ii) pre-processing with plain cleaning, and (iii) similarity scoring using Cosine similarity measure.

Analysis on timeline of a single user. Figure 5 shows correlation matrices representing similarities between pairs of bird songs of a single user. Figure 5a shows a matrix on the bird songs of *Barack Obama*. It highlights three blocks of highly correlated period of Tweets. The larger block [A] at the upper left corner represents Obama tweets during US presidential election in 2012. With a maximum Cosine similarity score of 0.33, it is clear that he has been tweeting a lot on a few correlated topics (voting, Romney, convention, health, etc. being among the most recurrent top terms). Block [B] refers to periods when Obama was interested in fiscal cliff. Finally, block [C] relates to the shooting in the Newtown elementary school, during which Obama's major topic terms were gun, violence, and weapon.

The correlation matrix in Fig. 5b shows an interesting behavior of a normal "user D" (as opposed to a public figure or news media). As discussed in Sect. 4.3, mentions are very frequent among common users. Analyzing user D's bird songs without filtering out mentions, the plot highlights two blocks, [A] and [B]. The similarity of bird songs are dominated by the use of mentions to particular follower/followee of his. Investigating key terms in the time period of block [A], user D was exchanging messages with one of his follower. After one week of pause, in block [B], user D then mentions about another follower of his (and never refers to the follower in [A]). We suppose that user D's sudden change in his mentions indicates a change in his social relationship, e.g., change of his dating partner.

Moreover, Fig. 5c shows a typical correlation matrix of generic "normal users". Compared to a public figure's correlation matrix (Fig. 5a), the size of correlated blocks is small and more uniform. Likewise, the maximum similarity score is also lower at 0.26. This can be explained by different use of Twitter between public figures and normal users; public figures use Twitter to deliver messages with substantial topics [2, 6], whereas normal users use Twitter to socialize (with messages on status updates, social signals, messages indicating mood, etc.) as noted in [6].

Lastly, Fig. 5d shows a typical correlation matrix of a news media profile (CNN breaking news tweets). Given the behavior of this type of user, $T = 7$ is too long because it is very unlikely, for a breaking news, that a fine grained topic last more than one week. Hence, we set $T = 2$ in order to highlight the most recurrent topics among the news articles. A total of six blocks of highly correlated period emerge, which represent topic of news lasting more than 2 days. Looking at the common topic

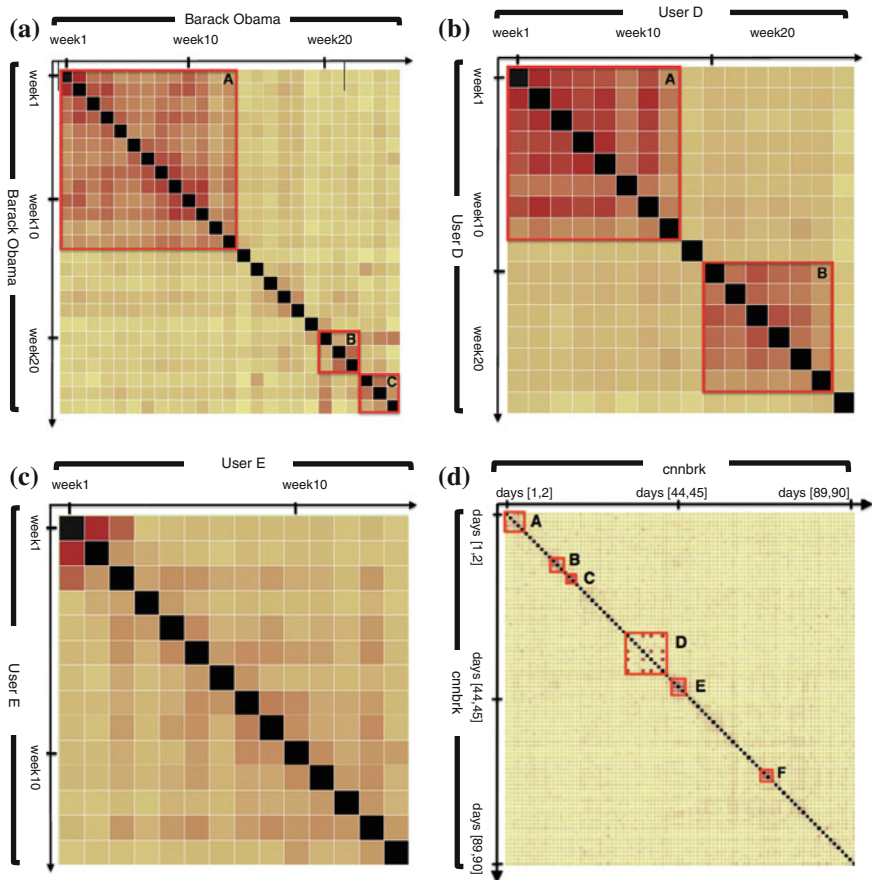


Fig. 5 Similarity among bird songs for different type of users. $T = 7$ days (**a**, **b**, **c**) and $T = 2$ days (**d**), plain cleaning, Cosine similarity. **a** Barack Obama-Max similarity = 33 %. **b** User D-Max similarity = 31 %. **c** User E-Max similarity = 26 %. **d** cnnbrk-Max similarity = 64.18 %

terms, Block [A] refers to the olympic games and block [B] relates to tropical storm Isaac. Following, a coarser correlated period represented by block [C] focus on the US election period. The last two blocks refers to Hurrican Sandy and shooting in Newtown school, respectively.

Finally, TUCAN can also be instrumented to highlight artificial similarity among a user's tweet that were generated by automatic tools like Foursquare check-in, auto-tweet tools, etc. We do not report their examples for sake of brevity.

Analysis across different users. Besides per-user analysis, TUCAN can infer semantic relationships across a multiple of users when applied to a group of target users. We select ten public figures and media blogs and report the cross-similarity matrix in Fig. 6. The latest six bird songs with $T = 14$ days are considered, referring to a

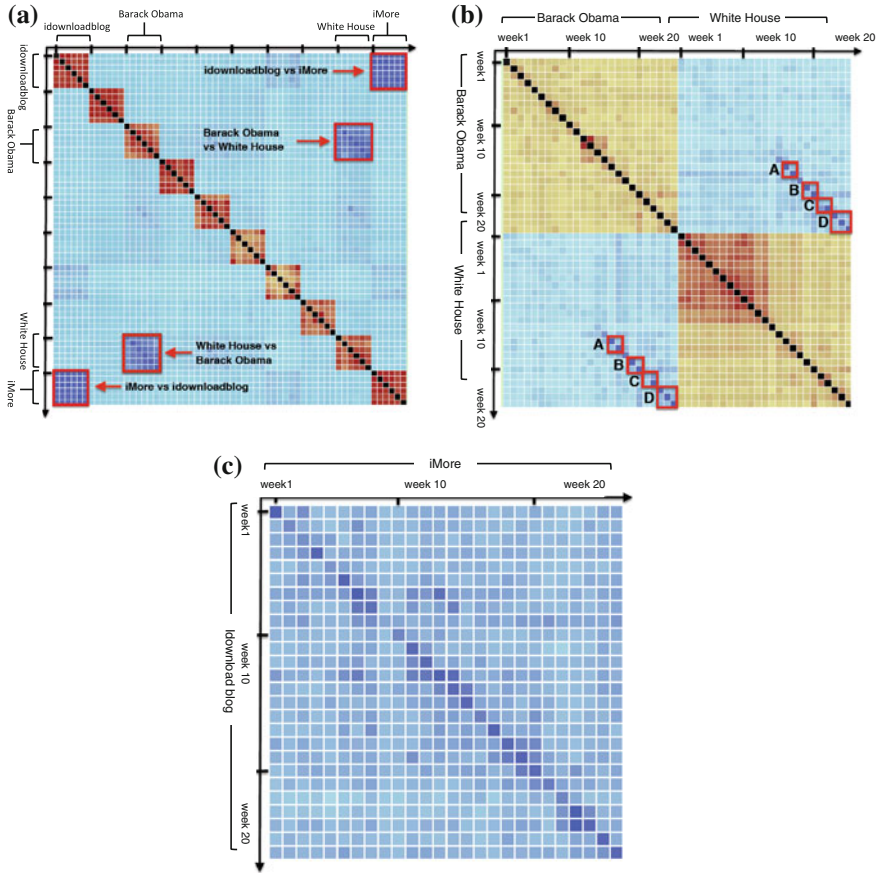


Fig. 6 Similarity among users over different bird songs. Plain cleaning and Cosine similarity. **a** Famous users versus famous users. $T = 14$ days. **b** Barack Obama versus White House. $T = 7$ days. **c** idownloading versus iMore. $T = 7$ days

common period of time. Each bird song is checked against each other. Results are represented as a colored matrix, using different color scales (and normalization) for blocks outside the main diagonal and in the main diagonal (where same-user's bird songs are compared). Focusing on the former, two pairs of users emerge as mostly correlated: {*Barack Obama*, *White House*} and {*idownloadblog*, *iMore*}.

Zooming in and increasing the resolution by selecting $T = 7$ days, Fig. 6b compares {*Barack Obama*, *White House*} in detail over 25 weeks of tweeting. First, notice that during Barack Obama's campaign (ref. Figure 5a) the correlation with White House is marginal. After elections, four periods of high correlations are pinpointed, highlighting the periods Barack Obama and White House publicize similar topics. The block [A] indicates the period of educational cost cut. [B] indicates the massacre at Newtown. [C] refers to fiscal cliff, and [D] on reformation of US

immigration laws. The discovery of both well-correlated and non-correlated periods allows us to quantify periods of time the President spoke for himself (and his political party) and the government of the US.

Similar consideration holds when zooming in {*idownloadblog*, *iMore*} comparison in Fig. 6c. Both users are blogs reporting news on Apple products. Also in this case $T = 7$ days, for 25 bird songs. Only the cross-similarity macro block is shown for the sake of brevity. Notice the large similarity in the main diagonal; it indicates that the two profiles report the same news, whose duration last for short period of time. The behavior is justified by the fact that both accounts work as sources of technology news.

In summary, Fig. 6 shows how TUCAN can be leveraged to infer semantic relationships across an arbitrary set of users. Apart from the existence of topical correlations between two target users, one can derive: (i) how many similar topics they share, (ii) if the shared topics are in the same time periods or in different times, and (iii) what keywords these common topics share. Figure 6b shows that while *Barack Obama* and *White House* have less commonality overall, they have a few strongly correlated topics. On the other hand, Fig. 6c shows how *idownloadblog* and *iMore* overall share lots of common topics, but there are no signs of significantly similar topics. Zooming on Fig. 6b, one can see that the common topics are shared during the same time periods.

5 Conclusion

In this paper we presented TUCAN, a framework to graphically represent semantic correlations of individual Twitter users' timelines. Building on text mining techniques, TUCAN analyses "bird songs", i.e., group of tweets belonging to the same time period, and compares their similarity. The analyst is offered a GUI to investigate the impact of different pre-processing and similarity definitions. Experiments conducted on actual Twitter users show the ability to pinpoint recurrent topics, or correlations among users.

There are several avenues for future work. First, we would like to expand our framework to be able to model patterns of topic durations and transitions. Leveraging the measurements revealing the correlation durations of topics, accumulating the statics for long-term can reflect changes in the user's interests. Second, we are interested in inferring users' social relationships based on their topical relations.

References

1. Java A, Song X, Finin T, Tseng B (2007) Why we Twitter: understanding microblogging usage and communities. In: Workshop on web mining and social network, analysis, pp 56–65
2. Kwak H, Lee C, Park H, Moon S (2010) What is Twitter, a social network or a news media? In: WWW, pp 591–600

3. Alvanaki F, Michel S, Ramamritham K, Weikum G (2012) See what's enBlogue—real-time emergent topic identification in social media. In: EDBT. ACM, Berlin
4. Hong L, Davison BD (2010) Empirical study of topic modeling in Twitter. In: Workshop on social media analytics. ACM, New York, pp 80–88
5. Mathioudakis M, Koudas N (2010) Twittermonitor: trend detection over the Twitter stream. In: SIGMOD'10. ACM, New York, pp 1155–1158
6. Ramage D, Dumais ST, Liebling DJ (2010) Characterizing microblogs with topic models. In: Cohen WW, Gosling S (eds) ICWSM. The AAAI Press
7. Salton G, Mcgill MJ (1986) Introduction to modern information retrieval. McGraw-Hill Inc, New York
8. Grimaudo L, Song H, Baldi M, Mellia M, Munafò M (2013) TUCAN Twitter user centric analyzer. In: Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining
9. Ramage D, Hall D, Nallapati R, Manning CD (2009) Labeled LDA: a supervised topic model for credit attribution in multi-labeled corpora. In: Proceedings of the 2009 conference on empirical methods in natural language processing: volume 1—volume 1. Stroudsburg, pp 248–256
10. Chang J, Boyd-Graber J, Blei DM (2009) Connections between the lines: augmenting social networks with text. In: ACM SIGKDD. ACM, New York, pp 169–178
11. Zhao WX, Jiang J, Weng J, He J, Lim EP, Yan H, Li X (2011) Comparing twitter and traditional media using topic models. In: ECIR'11. Berlin, pp 338–349
12. Phan X-H, Nguyen L-M, Horiguchi S (2008) Learning to classify short and sparse text and web with hidden topics from large-scale data collections. In: WWW. New York, pp 91–100
13. Liu Y, Niculescu-Mizil A, Gryc W (2009) Topic-link LDA: joint models of topic and author community. In: Annual international conference on machine learning. ACM, New York, pp 665–672
14. Blei DM, Ng A, Jordan M (2003) Latent dirichlet allocation. JMLR 3:993–1022
15. Rosen-Zvi M, Griffiths T, Steyvers M, Smyth P (2004) The author-topic model for authors and documents. In: UAI. Arlington, pp 487–494
16. Rosen-Zvi M, Chemudugunta C, Griffiths T, Smyth P, Steyvers M (2010) Learning author-topic models from text corpora, vol 28(1). ACM, New York, pp 4:1–4:38
17. Das Sarma A, Jain A, Yu C (2011) Dynamic relationship and event discovery. In: WSDM. New York, pp 207–216
18. Malik S, Smith A, Hawes T, Papadatos P, Dunne C, Shneiderman B (2013) TopicFlow: visualizing topic alignment of twitter data over time. In: Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining
19. Porter M (1980) An algorithm for suffix stripping. Program 14(3):130–137
20. Fellbaum C (1998) WordNet: An Electronic Lexical Database. MIT Press, Cambridge, p 422
21. Honeycutt C, Herring SC (2009) Beyond microblogging: conversation and collaboration via Twitter. In: HICSS'09, pp 1–10

Evaluating Important Factors and Effective Models for Twitter Trend Prediction

Peng Zhang, Xufei Wang and Baoxin Li

Abstract Trend prediction for social media has become an important problem that can find wide applications in various domains. In this work, we investigate two basic issues in Twitter trend prediction, i.e., what are the important factors and what may be the appropriate models. To address the first issue, we consider different content and context factors by designing features from tweet messages, network topology, and user behavior, etc. To address the second issue, we investigate several prediction models based on combinations of two fundamental model properties, i.e. (non-)linearity and (non-)state-space. Our study is based on a large Twitter dataset with more than 16 M tweets and 660 K users. We report some insightful findings from comparative experiments. In particular, it is found that the most relevant factors are derived from user behavior on information trend propagation and that non-linear state-space models are most effective for trend prediction.

Keywords Social media · Trend prediction · Twitter · Temporal modeling

1 Introduction

An information trend is the process of the information on some underlying topic or event propagating with noticeable distinction through network links. Being able to predict or simulate the outcomes of such a process may lead to many applications in domains such as politics, economics and business [1, 4]. A survey of applications based on information trend prediction using social media can be found in [25]. In this study, we use the Twitter network [9] as a case study and focus on the prediction of hashtag trend. ***Twitter*** is an interactive social media platform where users share

P. Zhang · X. Wang · B. Li (✉)
Computer Science and Engineering, Arizona State University, Phoenix, USA
e-mail: baoxin.li@asu.edu

P. Zhang
e-mail: pzhang41@asu.edu

X. Wang
e-mail: xufei.wang@asu.edu

J. Kawash (ed.), *Online Social Media Analysis and Visualization*,
Lecture Notes in Social Networks, DOI 10.1007/978-3-319-13590-8_5

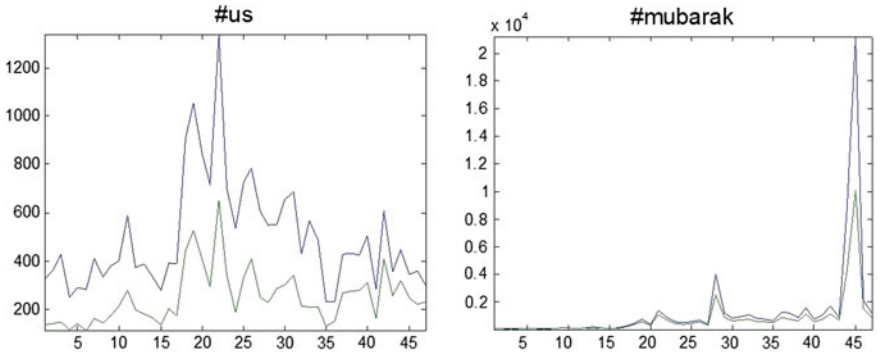


Fig. 1 Illustration of hashtag trends. The *horizontal axis* is time, and *vertical axis* is count. In each plot, the *blue*/*green line* is the number of tweets/users respectively

ideas by ***tweets*** which are messages of less than 140 characters. The term ***hashtag trend*** in Twitter refers to the dynamics of a set of tweets grouped by a hashtag, which is a string starting with the character #, to represent a topic, event, etc. For example, #ASONAM13 is the hashtag about the ASONAM 2013 conference. In addition to hashtag trend, other types of trends exist in Twitter, such as the trend related to a special event rather than a hashtag, to which we use Twitter trend to refer. We focus on hashtag trend, because it is easy to extract via hashtag string, and also the diffusion of hashtag trend may share some common traits with general Twitter trends. A hashtag trend is measured by the number of users and tweets involved in each time interval (Fig. 1). Accordingly, our objective of prediction is defined as: Given the history of a hashtag trend, to predict how many users and tweets will appear in the next time interval.

Hashtag trends may evolve with different patterns, as illustrated in Fig. 1, due to many interacting factors. Designing trend-defining features and adopting a proper model are two key tasks in effective trend prediction. Although trend prediction has been studied in other related fields, e.g., system identification [14], important questions are still open in the context of social networks. For example, trend diffusion is complex due to the interaction among a large set of network nodes, which requires sophisticated models. Also, how to exploit additional factors such as user behavior in social networks should be investigated. Hashtag trend prediction provides a rich problem for investigating all these issues, as successful prediction of hashtag trend may help us to better understand the relevant factors and appropriate models of trend diffusion, which may also benefit other applications like hashtag recommendation and trend detection

Current research on tweeter trend prediction mainly focuses on relevant trend factors. Such factors identified in the existing work typically fall into one of the two categories: context factors or content factors. Content factor is derived from tweets to describe the trend topic, quality, etc. Context factor is derived from network structure and user behavior to describe the environment for trend diffusion. In terms of prediction models, existing methods typically use only simple regression or classification

models, which are in general inadequate for handling complex trend dynamics on social networks. It remains to be fully explored to determine the best features or feature combinations and to design the most appropriate prediction models for trend prediction on Twitter. That is, we set out to answer the following two questions: what are the important factors and what may be the appropriate models.

In this work, to address the first question, we combine different content and context features for trend prediction and perform relevance analysis. The features are derived from the tweet content, network topology, and user behavior. Feature importance was individually identified in the literature without analyzing the effect of different combinations. Our analysis is meant to fill this gap in knowledge. To address the second question, we compare several types of prediction models, which are based on combinations of two fundamental model properties, i.e. (non-)linearity and (non-)state-space modeling. The validity of the features and models is evaluated through experiments on a large Twitter dataset.

The main contributions and findings of this study are as follows. First, we combine different aspects of relevant trend factors, i.e., content and context factors, for trend prediction and give comprehensive study on their importance and relevance. Second, different categories of prediction models, including (non-)linear and (non-)state-space models, are investigated and compared. Experiments on a large Twitter dataset suggest that trend factors based on user behavior is most useful, and nonlinear state-space models appear to be more effective. Although our study is on hashtag trend, it may be helpful to analyzing other social media trends due to the common nature of information diffusion on social networks.

The remaining of this chapter is organized as follows. In Sect. 2, we briefly review related work on both relevant factors and prediction models for Twitter trend. Based on the notations and problem statement in Sect. 3, we introduce our trend factors and prediction models in Sects. 4 and 5 respectively. Experiments and discussion are given in Sect. 6. We finally conclude in Sect. 7.

2 Related Work

We briefly review the literature on relevant trend factors and trend prediction models in the following subsections.

2.1 On Relevant Trend Factors

Current research has identified many factors relevant to Twitter trend prediction. These factors can be generally divided into two categories, i.e. content and context factors.

Content factors describe the properties of a trend through lexical, semantic, and sentimental analysis. For example, it was recognized in [13, 18] that trends may

follow different temporal patterns depending on the underlying topics (e.g., politics, sports, etc.). A LDA topic distribution model was used to predict trends in [8]. The TF-IDF similarity between the information content and the users interest is also found relevant [23]. Sentimental features based on the content of tweets are used to predict movie box-office performance [1] and stock price [4]. There are also simple content features, such as the fraction of tweets containing URL [15], the fraction of retweet/mention in a trend [10], and features derived from hashtag itself [21].

Context factors describe the environment in which the trend is propagating. On one hand, network topology has been shown to be related to the scale and speed of the diffusion trend. For example, network density was studied in [11] and the border of a sub-graph formed by users who already adopted the trend was studied for trend prediction in [18]. On the other hand, the importance of user behavior is also recognized. For example, the authors of [2] pointed out that the number and influence of information contributors, and portion of retweet users have a strong positive correlation with trend popularity. The importance of mention rate and retweet rate of information contributors is also recognized in [22].

On the micro level, there are also studies on the behavior of individual users, e.g. retweet [12] or hashtag adoption [24]. In general, the findings are similar to those of macro level Twitter trend prediction, and thus they also provide inspirations for designing features for trend prediction. In this study we generalized the attention limit of a user in [12] to design the user stimulus as a node context factor for trend prediction (Table 1).

Although many factors have been analyzed in the literature, most existing studies perform trend prediction with only one type of factors, except maybe two recent efforts [15, 21], which still covered only a very limited set of context and content features. In this study, different factors are combined together for trend prediction, and their importance is comprehensively analyzed with a large real dataset.

2.2 On Trend Prediction Models

The most popular trend prediction method is regression or classification [15, 21]. A majority of existing work relies on linear regression models due to its simplicity and effectiveness. However, the temporal history, which has proven to be important for trend prediction [23], is ignored. Another type of methods is to model information trends by differential stochastic equations [2, 10, 16, 20]. These methods explicitly describe the trend dynamics. However, due to too many relevant trend factors, such methods have to adopt assumptions that heavily simplify trend factors to make the problem tractable. There are also studies on the temporal trend pattern for popularity prediction [6, 13, 23]. Usually, the trends are grouped by their temporal patterns, and a classifier is assigned to each group based on content or context features. These methods can predict the trend in a whole range rather than the next time interval. However, it is not easy to update the prediction with the upcoming observations. Also, such methods will not provide any insights on the dynamic mechanism of information

Table 1 Summary of content and context trend features

Index	Description
Content factor	
1	Portion of retweet (rtT_h) in T_h: ($\|rtT_h\|$)/($\|T_h\|$) (Eq. (3))
2	Portion of mention (mT_h) in T_h: ($\|mT_h\|$)/($\|T_h\|$) (Eq. (3))
3	Portion of new tweets (nT_h) in T_h: ($\|nT_h\|$)/($\|T_h\|$) (Eq. (3))
4	Portion of URL tweets ($urlT_h$) in T_h: ($\|urlT_h\|$)/($\|T_h\|$)
5	Portion of trend users (oU_h) in U_h: ($\|oU_h\|$)/($\|U_h\|$) (Eq. (4))
6	Portion of trend followers (fU_h) in U_h: ($\|fU_h\|$)/($\|U_h\|$) (Eq. (4))
7	Portion of self-motivated users (sU_h) in U_h: ($\|sU_h\|$)/($\|U_h\|$) (Eq. (4))
8	Portion of oT_h as Tweets posted by oU_h: ($\|oT_h\|$)/($\|T_h\|$) (Eq. (5))
9	Portion of fT_h as Tweets posted by fU_h: ($\|fT_h\|$)/($\|T_h\|$) (Eq. (5))
10	Portion of sT_h as Tweets posted by sU_h: ($\|sT_h\|$)/($\|T_h\|$) (Eq. (5))
Structure context factor	
11	Ratio between border and trend users: ($\|bU_h\|$)($\|tU_h\|$) (Fig. 2)
12 $\sim$ 13	Max/Average of out-degree prestige of trend user tU_h. (Fig. 2)
14	Density of the sub-graph formed by tU_h. (Eq. (6))
15	Density of the bipartite graph between tU_h *and* bU_h. (Fig. 2)
16	Reciprocity of the sub-graph formed by tU_h. (Eq. (7))
17	Reciprocity of the bipartite graph between bU_h and tU_h
Node context factor	
18 $\sim$ 19	Average general activeness ($act(u;t)$) of tU_h or bU_h. (Eq. (8))
20	Average trend activeness ($act_h(u;t)$) of users in tU_h. (Eq. (8))
21	Ratio between the trend activeness and general activeness of tU_h
22 $\sim$ 23	Average trend stimulus ($stim_h(u;t)$) of tU_h or bU_h. (Eq. (9))
24 $\sim$ 25	Ratio between the trend and general stimulus of tU_h or bU_h
26 $\sim$ 27	Percentage of interaction among Tweets posted by tU_h or bU_h
28 $\sim$ 29	Ratio of interactions received by tU_h or bU_h

In this table, time index t is usually ignored for brevity

trend. In this study, we follow the line of regression model for Twitter trend prediction. However, we go beyond simple linear regression by employing dynamic state-space models in which the influence of temporal history is incorporated through latent state variables. We also investigate non-linear models to explore the complex interaction among relevant trend factors.

3 Notations and Problem Statement

Twitter network is a directed graph $G = \langle U, E \rangle$ where users U are connected to each other by social links E. A link from user u_i to u_j indicates that u_i is a ***follower*** of u_j, and u_j is a ***friend*** of u_i. For u_i, the number of friends is its ***out-degree*** and the

number of followers is its ***in-degree***. A users tweets will be broadcasted to all his followers, so links are the basic pipelines for information diffusion. Users can also contact each other via retweet and mention, which are the main cause of information diffusion in Twitter. ***Retweet*** is identified by the adoption of "RT @username" or "via @username" in a tweet; ***Mention*** is the use of "@username" if it is not a retweet.

For notions, uppercase letters U, T and their variations by prefix and subscripts represent a set of users and tweets. The lowercase letter u is for a user, h is for a hashtag trend, and t is for a time interval. For example, $T_h(t)$ is the set of tweets with hashtag h posted in time t, and $U_h(t)$ is the set of users who posted at least one tweet in $T_h(t)$. Let $tU_h(t)$ be the trend users of trend h in time t, the set of users who already adopted h in t, i.e. the $tU_h(t) = \bigcup_{\tau=1}^{t} U_h^{\tau}$. The trend border $bU_h(t)$ is the set of followers of $tU_h(t)$ who still have not adopted h yet. The notation of set cardinality is $|\cdot|$. The time index t is often omitted for brevity when no confusion arises.

Trend prediction is based on the ***popularity measure*** of trend h, which is defined as the number of tweets and users involved in time t:

$$\boldsymbol{y}_h(t) = [\log(|T_h(t)|), \log(|U_h(t)|)], \tag{1}$$

where the logarithm is used to compress the large dynamic range. Let $\boldsymbol{x}_h(t)$ be the vector of relevant trend factors. Then $\boldsymbol{y}_h(t)$ is estimated by a prediction model $\mathscr{F}(\cdot|\boldsymbol{\theta}|)$ as:

$$\hat{\boldsymbol{y}}_h(t+1) = \mathscr{F}(\boldsymbol{y}_h(1:t), \boldsymbol{x}_h(1:t)|\boldsymbol{\theta}), \tag{2}$$

where $\boldsymbol{\theta}$ is the parameter vector. $\boldsymbol{y}_h(1:t)$ and $\boldsymbol{x}_h(1:t)$ are the popularity measures and relevant trend factors up to current time t. The ***trend factors*** $\boldsymbol{x}_h(t)$ in this study are summarized in Table 1, which are derived from tweet content, network topology, and user behavior. As for $\mathscr{F}(\cdot|\boldsymbol{\theta})$, we investigate several models in Table 5 from combinations of two fundamental properties: (non-)linearity and (non-)state-space.

4 Relevant Trend Factors

In this section, we design features for relevant trend factors of both the content and the context group.

4.1 Content Factors

Content factors are extracted from tweets to describe the properties of the trend (e.g., topic and quality). In this study, we focus on simple content features such as the number of retweet and mention, without sophisticated semantic analysis. Despite of their simplicity, such content factors has proven to be useful for trend prediction [2]. Although semantic content factors are also relevant to trend diffusion [18], they are difficult to reliably extract. Traditional semantic analysis, e.g. Latent Dirichlet

Allocation, only leads to less effective features than the simple content factors [15]. Thus this study only focuses on simple content factors.

The trend popularity of Eq. 1 is defined on both the tweet set $T_h(t)$ and the user set $U_h(t)$, which can respectively be further divided into subsets for analysis, as shown below.

One simple division of $T_h(t)$ is by the type of tweets. From Sect. 3, $T_h(t)$ can be categorized into three subsets: retweet ($rtT_h(t)$) as the propagation of the old messages; mention ($mT_h(t)$) as the current discussion among trend users; and the remaining ($nT_h(t)$) as tweets with new information. That is, we can write:

$$T_h(t) = rtT_h(t) + mT_h(t) + nT_h(t). \tag{3}$$

The above three subsets play different roles in trend diffusion. A large $rtT_h(t)$ means the trend is in fast propagation; a large $mT_h(t)$ indicates the trend is under hot discussion among users and may show some stickiness to users; a large $nT_h(t)$ implies a lot of new information may be injected into trend h.

Based on the role of a users contribution, we can also divide $U_h(t)$ into the following 3 subsets. The first ($oU_h(t)$) consists of "old" users reposting trend h in t, i.e. $U_h(t) \cap tU_h(t-1)$ (Fig. 2). The remaining two groups are new trend users with different relation to $tU_h(t-1)$. Specifically, the second subset ($fU_h(t)$) are either the followers of $tU_h(t-1)$ or users retweeting trend from $tU_h(t-1)$. The third subset ($sU_h(t)$) are "self-motivated" users who either publish trend tweets by themselves or retweet from users outside our network (considering we can only crawl a subset of the entire Twitter network). In summary, we have:

$$U_h(t) = oU_h(t) + fU_h(t) + sU_h(t) \tag{4}$$

The three user groups in Eq. 4 play different roles in trend diffusion: $oU_h(t)$ implies the trends consistency by the activeness of old users, $fU_h(t)$ indicates the trends propagation through its borders, and $sU_h(t)$ suggests the trends growth by the information from outside the network.

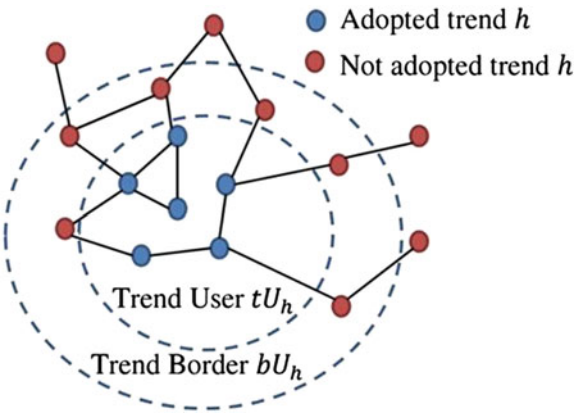


Fig. 2 Illustration of trend users (tU_h) and trend border users (bU_h)

Following the user categorization of Eq. 4, the tweeter set $T_h(t)$ can also be categorized by its producers as:

$$T_h(t) = oT_h(t) + fT_h(t) + sT_h(t), \tag{5}$$

where $oT_h(t)$, $fT_h(t)$, and $sT_h(t)$ are the tweet sets posted by the users in $oU_h(t)$, $fU_h(t)$, and $sU_h(t)$ respectively.

We also considered the subset of $T_h(t)$ with URL ($urlT_h(t)$). Due to the limited characters of tweets, the existence of URL suggests abundant content in a tweet. In fact, tweets with URL are more likely to be retweeted.

4.2 Context Factors

Context factors describe the network environment in which information trends are diffusing. The environment can be either the network topology (structure context) or the attributes of network nodes (i.e., users) (node context). Usually, a trend only exits on a set of interest users. For the sake of efficiency, only two user sets are considered here. The first is trend users tU_h (Fig. 2), because tU_h are the producers of trend h. The second one is the trend border bU_h, since bU_h, as the followers of tU_h, are the users directly exposed to the trend h. The related factors and their feature design are detailed below.

Structure context factors are metrics on the topological structure of network [17]. Three sub-graphs are considered, i.e. the sub-graph of tU_h, the sub-graph of bU_h, and the bipartite graph between bU_h and tU_h. The links between bU_h and tU_h are the major pipelines of trend propagation to new users. For each of the three sub-graphs, the network topology is described by three factors, i.e. centrality, density, and reciprocity, whose definition for a directed graph $G = \langle U, E \rangle$ is given below. The centrality [17] of G is measured by the average out-degree of its nodes. Centrality measures the users' influence for trend diffusion. The link density [17] of G implies its capacity of information diffusion as:

$$density(G) = |E|(|U| \cdot (|U| - 1)), \tag{6}$$

which is the number of links divided by all possible links. The reciprocity describes the tie-strength among users and by the portion of co-links in G:

$$reciprocity(G) = |coLink(E)||E|, \tag{7}$$

where $coLink(E) \in E$ is the subset of co-links [17].

Node context factors describe the users by their behavior and profile. Similar to structure context factors, the node context features will be derived from the two most relevant user sets, i.e. tU_h and bU_h. Details are given below.

Activeness is a users frequency to generate information, i.e. tweets. The ***general activeness*** of a user u on time t is:

$$act(u; t) = \alpha \cdot act(u; t - 1) + |T(u; t)|, \tag{8}$$

where α is the decay coefficient and $T(u;t)$ is the tweets posted by u on t. Trend activeness ($act_h(u;t)$) is defined similar to Eq. 8 by replacing $T(u;t)$ with $T_h(u;t)$, the set of tweets with hashtag h posted by u on t. A large $act_h(u;t)$ implies a high probability of a user's participation in h, and the ratio between $act(u;t)$ and $act_h(u;t)$ reflects the specific interest of u on the trend h. The activeness of a user set, e.g. tU_h and bU_h, is the average of the metric over its users.

Stimulus is the volume of information, i.e. tweets, received by a user. The ***general stimulus*** of user u on time t is the number of tweets posted by his friends:

$$stim(u;t) = \beta \cdot stim(u;t-1) + \sum_{u_i \in friend(u)} |T(u_i;t)|, \tag{9}$$

where β is the decay coefficient, $friend(u)$ the friends of u, and $T(u;t)$ the tweets posted by u on t (same as in Eq. 8). Similarly to Eq. 9, trend stimulus ($stim_h(u;t)$) is defined by replacing the $T(u;t)$ by $T_h(u;t)$, the set of tweets with hashtag h posted by u on t. $stim_h(u;t)$ reflects the volume of trend information received by u. The ratio between $stim_h(u;t)$ and $stim(u;t)$ reflects u's attention on a specific trend h. It was shown in [12] that the prediction of hashtag adoption by a user can benefit from attention-limited similarity measure. The general/trend activeness of a user set, e.g., tU_h or bU_h, is the averaged version over its element users.

Besides the above activeness and stimulus as the measure of user output and input information, user profile and action style are also relevant. In fact, user interaction, i.e. retweet and mention, is the main course of information diffusion in Twitter. Two factors are considered here. The first factor is the percentage of interactions among all tweets posted by a user up to time t, which reflects the willingness of a user to spread information. The second factor is the number of interaction tweets received by a user divided by all his tweets up to t, which reflects his influence in the network [19]. The two factors will be extracted on both the user set of tU_h and tU_h. The above content factors and context factors for our trend prediction are summarized in Table 1.

5 Trend Prediction Models

In this section, we introduce the prediction models used in this study. The details of the models are presented after the introduction of some preliminaries.

5.1 Preliminaries

A prediction model in Eq. 2 is a mapping from the input variables to the predicted objects. Following the convention of system identification [14], we use $\boldsymbol{y}(t)$ for the output vector under prediction, and $\boldsymbol{u}(t)$ for the input vector. For state-space model,

the state vector is $\boldsymbol{x}(t)$. ***Non-state-space*** models predict the next output $\boldsymbol{y}(t+1)$ from several past input and output:

$$\begin{aligned}(y(t+1) &= f(\boldsymbol{\varphi}_{n_a,n_b,n_k}) \\ \boldsymbol{\varphi}_{n_a,n_b,n_k} &= [\boldsymbol{y}(t-(0:n_a-1)), \boldsymbol{u}(t-(n_k:n_k+n_b-1))],\end{aligned} \tag{10}$$

where $f(\cdot)$ is the prediction function, n_k the input delay, and the model order is n_a and n_b. Non-state-space models are popular due to their simplicity. However, the prediction is limited to previous n_a output and n_b input in $\boldsymbol{\varphi}_{na,nb,nk}$. The history beyond the model order is ignored.

State-space models solve the limited-memory problem of non-state-space models by introducing a state vector $\boldsymbol{x}(t)$ as:

$$\begin{aligned}\boldsymbol{x}(t+1) &= h(\varphi_{n_a,n_b,n_k}, \boldsymbol{x}(t-n_d:t)) \\ \boldsymbol{y}(t+1) &= g(\varphi_{n_a,n_b,n_k}, \boldsymbol{x}(t+1-n_d:t+1)),\end{aligned} \tag{11}$$

where $\boldsymbol{\varphi}_{n_a,n_b,n_k}$ is the same as Eq. 10, $h(\cdot)$ the state-transition function, $g(\cdot)$ the output function, and n_d the state order. State-space models can predict with all history, as the effect of previous observations is accumulated in $x(t)$. Equation 11 is a general form of state-space model, where $h(\cdot)$ and $g(\cdot)$ can be either linear or non-linear with many different implementations as introduced below.

5.2 Prediction Models

ARX model (Auto Regressive model with eXternal input) [14] is a linear non-state-space model defined as:

$$\boldsymbol{y}(t) = \sum_{j=1}^{n_b} \boldsymbol{B}_j \cdot \boldsymbol{u}(t-n_k-j) - \sum_{i=1}^{n_a} \boldsymbol{A}_i \cdot \boldsymbol{y}(t-i), \tag{12}$$

where $\boldsymbol{A}_i$ and $\boldsymbol{B}_j$ are parameter matrices. The input vector can be ignored by setting $n_b = 0$, leading to the ***AR*** model. ARX can be estimated with simple least square method which makes it the most popular baseline model [14].

Nonlinear ARX (NARX) model (Fig. 3) is a non-linear non-state-space model implemented by feed-forward neural network. The nonlinearity lies in the neurons sigmoid transfer function operating on linear combination of layers output. NARX is trained by back-propagation. With enough layers and hidden nodes, NARX can approximate any function well [8].

Random forest (RF) [7] is a popular ensemble non-linear non-state-space method as a set of regression trees. The robustness of RF lies in the independence among different trees which is achieved in two steps. First, each tree is constructed independently by bootstrapping. Second, each tree node split is decided on a random subset of the predictor variables, i.e. φ_{n_a,n_b,n_k} in Eq. 10. The final prediction result of RF is the average of each regression tree.

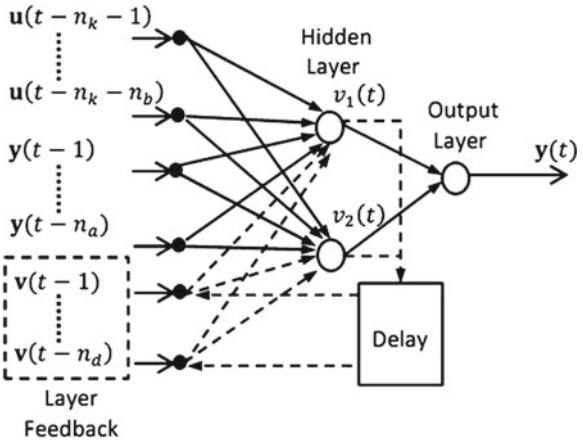


Fig. 3 Illustration of NARX and RNARX network with 1 hidden layer of 2 nodes. RNARX is the extension of NARX with the *dashed* feedback loop

Linear dynamic system (LDS) is a state-space model with linear state transition (h) and output emission (g) and $n_a = 0$, $n_b = n_k = n_d = 1$. The LDS can be formulated as:

$$\begin{aligned} \boldsymbol{x}(t+1) &= \boldsymbol{A} \cdot \boldsymbol{x}(t) + \boldsymbol{B} \cdot \boldsymbol{u}(t) + \boldsymbol{w}(t) \\ \boldsymbol{y}(t+1) &= \boldsymbol{C} \cdot \boldsymbol{x}(t) + \boldsymbol{D} \cdot \boldsymbol{u}(t) + \boldsymbol{v}(t) \end{aligned} \tag{13}$$

where $\boldsymbol{v}(t) \sim N(0, \boldsymbol{R})$ and $\boldsymbol{w}(t) \sim N(0, \boldsymbol{Q})$ are zero mean Gaussian noise. The LDS model order is the dimension of the state vector $\boldsymbol{u}(t)$. Another popular state-space model is HMM [3], which, however, is not appropriate for regression due to its discrete state space vector.

Recurrent Nonlinear ARX (RNARX) network is a state-space extension of NARX by the introduction of delayed feedback loops around each layer except for the last output layer (Fig. 3). The layer feedback can improve the network capability of modeling complicated dynamics, since the influence of history will accumulate on hidden nodes (or state vectors) through the feedback loops.

6 Experiments and Discussion

6.1 Dataset and Preprocessing

Our Twitter dataset is collected for the event of Arab Spring. The collection involved some manually defined hashtags and geographic regions related to the following countries: Egypt, Libya, Syria, Bahrain and Yemen. Meanwhile, the Twitter network

Table 2 Statistics of Arab Spring Twitter dataset

Name	Value	Name	Value
Tweets	16,043,422	Retweet	5,336,868
URL ratio	40.33 %	Hashtag ratio	97.48 %
Users	666,168	Links	86,710,704
Mean friend	130.20	Reciprocity	19.9 %

among the related users is crawled by the Twitter API as well. We collected 16.8 million Tweets by 0.67 million users from February 1, 2011 to August 31, 2011 using the streaming API. The crawled Tweets during this course account for approximately 10 % of the all Tweets hosted by Twitter following the above hashtag and geographic constraints. The statistics of our dataset is summarized in Table 2. The collection process leads to a high hashtag ratio, which means most tweets contain at least one hashtag. Therefore, the dataset is appropriate for hashtag trend prediction.

We selected the 336 most popular hashtags with at least 5,000 related tweets for trend prediction experiments. With the high hashtag ratio in Table 2, the 336 hashtags sum up to 28,333,656 tweets, which are more than the total number of tweets 16,043,422 (Table 2) since a tweet might contain several hashtags. Basic statistics of these trends are illustrated in Fig. 4, while Fig. 5 gives more insight on the trend characters. For example, in some hashtag trends e.g. #tel4rent and #akharmasr, only a few users posted a lot of tweets, which might be advertisement. The most popular trends are the Arab Spring hashtags, e.g. #egypt and #libya, which have a large population of trend users and a high average tweets.

We evaluate the prediction results by Mean Square Error (MSE) which is widely adopted to assess the result of time series prediction (e.g. [21]). The MSE of each hash-tag trend is averaged as the final MSE to assess the prediction model, and the MSE for predicting the number of user ($\log(|T_h(t)|)$) and tweets ($\log(|U_h(t)|)$) in Eq. 1 are reported separately. All prediction results in this section are estimated by

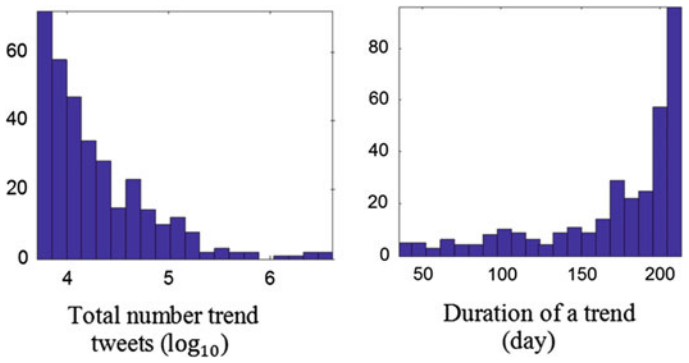


Fig. 4 *Left* The histogram of total number of tweets over all hashtag trends. *Right* The histogram of the trend duration over all hashtag trends

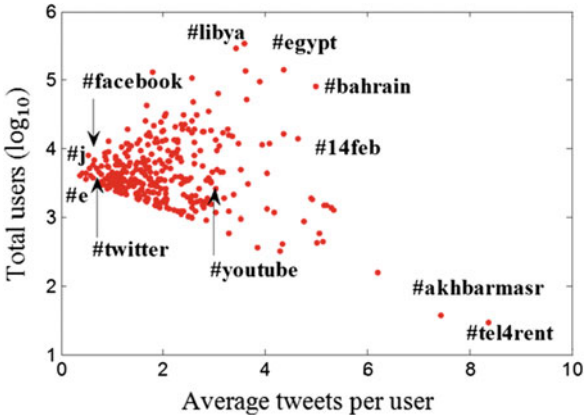


Fig. 5 The scatter plot of total users in a trend (*y-axis*) versus the average trend tweets posted by each user (*x-axis*)

10-fold cross-validation. The features in Table 1 are normalized before being fed into the prediction model.

6.2 Comparison of Trend Factors

With so many content and context factors listed in Table 1, several questions may arise: will these factors really help? which factor is most relevant for trend prediction? These questions are answered by the following feature importance and prediction accuracy analysis.

6.2.1 Factor Importance Analysis

We use random forest (RF) for feature importance analysis, which is the decay of test accuracy by permuting the feature value of out-of-bag samples [7]. The importance is scaled into [0,1] by dividing the highest value. We set the prediction vector φ_{n_a,n_b,n_k} to $n_a = n_b = 5$ and $n_k = 1$, which is the input and output value in the previous five days. The reason is that the trend prediction result by the RF model will not change significantly with a history longer than five days. In our study, the decay coefficient and for activeness (Eq. 8) and stimulus (Eq. 9) are set to 0.1. In fact, the result may not change too much for small decay coefficients, while large decay coefficients will degrade performance. Following the factor index of Table 1, we illustrate the mean variable importance of each factor over five days in Fig. 6. We only present the importance for the popularity measure of $U_h(t)$ because the result of $T_h(t)$ is similar. The statistics of the ten most important trend factors are listed in Table 3.

Based on the statistics in Fig. 6 and Table 3, we can see that the importance of node context factors, i.e. stimulus and activeness, are significantly higher than other trend

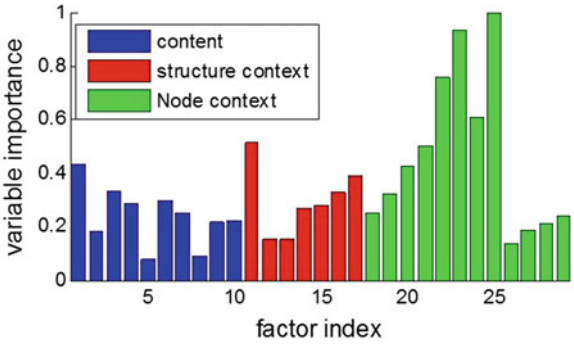


Fig. 6 The variable importance of different trend factors for user ($U_h(t)$) prediction. The factor index is in Table 1

Table 3 The 10 most important trend factors

Index	Category	log[$T_h(t)$]: Mean (Std)	log[$U_h(t)$]: Mean (Std)
22	Node context	1.000 (0.376)	0.917 (0.197)
23	Node context	0.788 (0.261)	0.964 (0.350)
24	Node context	0.744 (0.195)	1.000 (0.381)
25	Node context	0.883 (0.384)	0.722 (0.151)
21	Node context	0.560 (0.179)	0.559 (0.157)
11	Structure context	0.383 (0.029)	0.716 (0.076)
20	Node context	0.412 (0.139)	0.529 (0.098)
17	Structure context	0.336 (0.023)	0.560 (0.039)
1	Content	0.327 (0.184)	0.441 (0.166)
10	Content	0.332 (0.077)	0.346 (0.056)

The feature index is the same as Table 1. The features are sorted by the mean importance to predict $T_h(t)$ and $U_h(t)$

factors. On one hand, stimulus is a measure of information input to the users. A large trend stimulus means a high exposure to a trend, so the users are likely to join. On the other hand, activeness measures the trend information output by the users. Large trend activeness means a high action frequency, so the users are likely to post a trend tweet again. The second important type of trend factors is the structure context factors based on the relation between trend users (tU_h) and trend border (bU_h) (Fig. 2). This is also intuitive. For example, a high ($|bU_h|$)/($|tU_h|$) (Index 11, Table 1) means the trend are broadcast to a lot of users (bU_h), and a high reciprocity (Index 17, Table 1) means these listeners (bU_h) has a good relationship with the trend propagators (tU_h).

The above discussion leads to two conclusions. First, trend factors based on user behavior are more important (e.g., trend stimulus/activeness). This is consistent with the findings of [5] that the user interaction graph is more appropriate for influence measure. Second, the factors specially designed for trend diffusion are more important than those on general information. For example, the trend stimulus/

Table 4 MSE of prediction result from different trend factors

Factors	$\log[T_h(t)]$: Mean (Std)	$\log[U_h(t)]$: Mean (Std)
None	0.140 (0.009)	0.155 (0.008)
Content	0.136 (0.012)	0.151 (0.010)
Structure context	0.134 (0.010)	0.147 (0.009)
Node context	0.131 (0.008)	0.146 (0.009)
All	0.130 (0.010)	0.142 (0.008)

"All" means prediction with all trend factors in Table 1; "None" means prediction without trend factors, i.e. $n_b = 0$ for the $\boldsymbol{\varphi}_{n_a,n_b,n_k}$ in (10)

activeness and topological structures between bU_h and tU_h (Fig. 2) are more important than the general factor on users interaction ratio.

6.2.2 Prediction Performance Analysis

We further investigated the importance of different types of factors by comparing their prediction performance (Table 4). For the sake of consistency, we used the RF prediction model with the same setup as in the above factor importance analysis.

From the prediction MSE in Table 4, we can draw the following conclusions. First, the trend factors (Table 1) really helps to improve trend prediction, because the MSE with all trend factors is significantly better than that without any factors. Second, the MSE of node context, structure context, and content factors coincide with the variable importance in Fig. 6. However, their performance gap is not significant. This might be because these factors are complementary to each other and they have their own way of contributing to the solution.

6.2.3 Comparison of Prediction Models

In this subsection, we investigate different types of models for trend prediction. In this study, a model is characterized by two aspects, i.e., linear/non-linear and state-space/non-state-space, leading to a combination of four model categories. For each category, at least one typical model is chosen, and the model should have connection to models of other category. To be specific, LDS (Eq. 13) is the direct state-space extension of ARX (Eq. 12), and NARX (Fig. 3) is the direct non-linear extension of ARX. Moreover, RNARX is the state-space extension of NARX. So comparison among these models can reflect the effect of different model properties. The time complexity of the models was not of concern in this study since any of them has only nominal prediction complexity after the model is trained (and the training can be done offline).

The comparative prediction results are summarized in Table 5 with prediction MSE for evaluation. The prediction models were trained with all the features in Table 1. For each model, we tried a series of model setups, and present the best result. For ARX, RF, and NARX model, we went through order 1 to 5 for the predictor vector $\boldsymbol{\varphi}_{n_a,n_b,n_k}$ in Eq. 10. For LDS we checked the model order up to 4. For both the NARX

Table 5 Best prediction MSE of different models

Type	Model	log[$T_h(t)$]: Mean (Std)	log[$U_h(t)$]: Mean (Std)
L-NS	ARX	0.151 (0.009)	0.169 (0.006)
L-S	LDS	0.149 (0.009)	0.166 (0.007)
NL-NS	NARX	0.133 (0.007)	0.146 (0.008)
	RF	0.130 (0.010)	0.142 (0.008)
NL-S	RNARX	**0.128 (0.008)**	**0.139 (0.009)**
Baseline	Last	0.175 (0.015)	0.198 (0.018)
	Mean	0.219 (0.017)	0.235 (0.016)

L and *NL* is the short for linear and non-linear; *S* and *NS* is the short for state-space and non-state-space. Last and Mean are two baseline prediction methods, i.e. Last prediction and mean predictor, defined above

and RNARX models, the layer of network varies from 1 to 2, and the hidden node number varies from 2 to 10. The predictor order of RNARX (Fig. 3), i.e. n_a and n_b, varies from 1 to 2, and the delay order n_d is fixed to 1.

To show the effectiveness of above models, we also designed two baseline prediction methods. The first is called Last Predictor, which simply predicts by the last value, i.e. $\hat{y}(t+1) = y(t)$. The second is called Mean Predictor, which simply predicts by the mean value up to the latest time, i.e. $\hat{\boldsymbol{y}}(t+1) = t^{-1}\sum_1^t \boldsymbol{y}(t)$. Their prediction results are also included in Table 5.

We have the following observations from Table 5. First, the Twitter hashtag trend is predictable on some degree, because the MSE of all prediction models are significantly better than the two baseline methods, i.e. Last Predictor and Mean Predictor. Second, nonlinearity is very important for the trend prediction model, which is clear from the performance gap between linear models, i.e. ARX and LDS, and their nonlinear counterparts, i.e. NARX and RNARX. The non-linearity may be mainly due to the complex dynamics in information trend diffusion. Third, state-space is helpful, but the benefit is smaller than nonlinearity. In fact, the performance gap between ARX and LDS is not significant. The situation is similar for RF and RNARX. The slight improvement may be due to the fact that long history information might not be as important for Twitter trend as it is for other trend, e.g. economics. In fact, for Non-state-space models, i.e. ARX and RF, the performance will not change significantly with more than 5-day history. Another possible explanation is that state-space models usually use local gradient training methods, which might not be competent for handling complicated cost functions due to high feature dimension. Further feature selection or regularization might help the situation.

7 Conclusion

We studied the two key aspects of trend prediction, i.e. important factors and appropriate models, with a focus on Twitter. We investigated different factors on both tweet content and network context for trend prediction. We also compared prediction

models as the combination of two fundamental model properties, i.e., (non)linearity and (non-)state-space modeling. Experiments on a large Twitter dataset led to the following observations. First, both content and context factors can help trend prediction. However, node context factors of user-specific behavior on a trend are most relevant. As for the prediction model, non-linear models are significantly better than their linear counterparts, which may be due to the complex information diffusion process in large social networks. Further, when properly employed, state-space models can help to improve prediction although only to a slight degree.

Future work includes two directions for further exploration. First, the semantics and sentiments of a trend is relevant to its diffusion process, and thus further investigation on feature design is worthwhile. Second, a network may evolve over time and thus adaptive models may be needed to account for this.

Acknowledgments The work is supported in part by the National Science Foundation (NSF) via Grant #1135616. All views and opinions are solely of the authors.

References

1. Asur S, Huberman BA (2010) Predicting the future with social media. In: International conference on web intelligence and intelligent agent technology
2. Asur S, Huberman BA, Szabo G, Wang C (2011) Trends in social media: persistence and decay. In: International AAAI conference on weblogs and social media (ICWSM)
3. Bishop CM (2006) Pattern recognition and machine learning. Springer, New York
4. Bollen J, Mao H, Zeng X (2011) Twitter mood predicts the stock market. J Comput Sci 2(1):1–8
5. Cha M, Haddadi H, Benevenuto F, Gummadi PK (2010) Measuring user influence in Twitter: the million follower fallacy. In: ICWSM
6. Chen GH, Nikolov S, Shah D (2013) A latent source model for nonparametric time series classification. In: NIPS
7. Hastie T, Tibshirani R, Friedman J (2009) The elements of statistical learning: data mining, inference, and prediction, 2nd edn. Springer, New York
8. Haykin S (2008) Neural networks and learning machines, 3rd edn. Prentice Hall, Upper Saddle River
9. Kwak H, Lee C, Park H, Moon SB (2010) What is Twitter, a social network or a news media? In: WWW
10. Lerman K, Hogg T (2010) Using a model of social dynamics to predict popularity of news. In: WWW
11. Lerman K, Ghosh R (2010) Information contagion: an empirical study of the spread of news on Digg and Twitter social networks. In: ICWSM
12. Lerman K, Intagorn S, Kang J, Ghosh R (2012) Using proximity to predict activity in social networks. In: WWW (companion volume)
13. Lehmann J, Gonalves B, Ramasco JJ, Cattuto C (2012) Dynamical classes of collective attention in Twitter. In: World wide web (WWW)
14. Ljung L (1998) System identification: theory for the user, 2nd edn. Prentice Hall, Upper Saddle River
15. Ma Z, Sun A, Cong G (2012) Will this #hashtag be popular tomorrow? In: SIGIR
16. Matsubara Y, Sakurai Y, Prakash BA, Li L, Faloutsos C (2012) Rise and fall patterns of information diffusion: model and implication. In: KDD
17. Newman ME (2010) Network: an introduction. Oxford University Press, Oxford

18. Romero DM, Meeder B, Kleinberg JM (2011) Differences in the mechanics of information diffusion across topics: idioms, political hashtags, and complex contagion on Twitter. In: WWW
19. Suh B, Hong L, Piroll P, Chi EH (2010) Want to be retweeted? Large scale analytics on factors impacting retweet in Twitter network. In: IEEE international conference on social computing
20. Szabo G, Huberman BA (2010) Predicting the popularity of online content. Commun ACM 53:80–88
21. Tsur O, Rappoport A (2012) Whats in a hashtag?: content based prediction of the spread of ideas in microblogging communities. In: WSDM
22. Yang J, Counts S (2010) Predicting the speed, scale, and range of information diffusion in Twitter: In: ICWSM
23. Yang J, Leskovec J (2011) Patterns of temporal variation in online media. In: WSDM
24. Yang L, Sun T, Mei Q (2012) We know what @you #tag: does the dual role affect hashtag adoption? In: WWW
25. Yu S, Kak S (2012) A survey of prediction using social media. arXiv:1203.1647v1

Rings: A Visualization Mechanism to Enhance the User Awareness on Social Networks

Shi Shi, Thomas Largillier and Julita Vassileva

Abstract Users of social network sites, such as Facebook, are becoming increasingly overwhelmed by the growing number of "friends" and the "tsunami" of updates generated by them. It is very easy to miss potentially interesting updates, if one does not frequently check her news feed. Also it is hard to get a sense of which friends are active and especially, which are passive or completely gone. Awareness of friends and friends' activities, or the lack of them, is important to build trusted social networks. However, the current social network sites provide very limited cues to support these two kinds of awareness. This paper proposes a method to visualize the activity level of friends. It creates a time- and an activity-pattern awareness for the user, as well as an awareness of the lurkers. It also offers options to filter activity streams regarding particular time periods and friends.

Keywords Social networks · Social visualization · Information filtering

1 Introduction

Social Network Sites (SNS) have experienced an explosive growth in recent years. There are more than 874 million active users on Facebook as of September 30, 2013 of which 727 million login daily.[1] A large proportion of these users share updates

[1] http://newsroom.fb.com/Key-Facts.

S. Shi · J. Vassileva
MADMUC Lab, University of Saskatchewan, Saskatoon, Canada
e-mail: shey.shi@usask.ca

J. Vassileva
e-mail: jiv@cs.usask.ca

T. Largillier (✉)
GREYC Université de Caen - Basse Normandie, Caen, France
e-mail: thomas.largillier@unicaen.fr

J. Kawash (ed.), *Online Social Media Analysis and Visualization*,
Lecture Notes in Social Networks, DOI 10.1007/978-3-319-13590-8_6

of their status with friends, including messages about their thoughts, their current location, links to interesting articles or videos, statements of activities (e.g. they have befriended other users, or the messages generated as a side effect of playing games). Such status updates will be called "social data" in this paper. A large amount of social data is generated every day, which triggers an information overload for users. For example, the average user on Facebook has 130 friends, but teenagers have more than 300.[2] These friends normally generate a rich stream of social data available to the user whenever she logs in to her homepage on Facebook. However, it is very easy to miss something important or interesting, if one has not logged in for one or two days, and a lot of updates were shared by her friends during this period. Also it is not easy to find out if a particular user has posted something recently, or who is generally active on Facebook and who is just a lurker. Therefore, it is necessary to provide a better way to organize and present social data to make users aware of the pattern of online social activities. Information visualization technology can provide effective approaches for presenting large amounts of data intuitively, which can help users to get insights into the data, discover patterns and find information of interest easily.

Facebook is currently the most popular social network. It provides the user with the opportunity to monitor and comment on social data from her friends, and share her own social data. However, Facebook does not provide a way for the users to easily find amongst her friends:

- who are active users,
- who post many popular updates and who do not,
- who was active recently and who some time ago,
- who posted recently but whose updates have been missed by the user,
- who stopped sending updates and became a lurker or stopped using Facebook altogether.

This paper proposes an intuitive and easy to understand visualization method that creates the needed awareness for the user about her social network on Facebook providing answers to these questions. Additionally, the implemented visualization application provides navigational and interactive methods to access posts of all the user's friends, so she can browse the social data shared by her friends more easily and selectively.

The rest of the paper is organized as follows: Sect. 2 presents an overview of existing social visualization tools. Section 3 describes the conceptual design and implementation of the proposed approach. The evaluation of the proposed tool is presented in Sect. 4. Section 5 concludes the paper.

[2] http://expandedramblings.com/index.php/by-the-numbers-17-amazing-facebook-stats/#.Uq4yaGRDtF8.

2 Related Work

In 2012, Facebook announced a search engine for the immense amount of data shared: Graphsearch. Graphsearch allows users (but not any users, only those with English language setting of their FB, after special request and a waiting period), to make deep queries of the entered preferences, interests, locations and posts of their friends to find answers to questions like: Who of my friends has been in New York, friends who have particular interests and live in particular location, etc. This is not an awareness tool, but a tool that allows users to treat their social nework's posts as a searchable database, in case they have specific queries.

Recently Facebook updated the presentation of its Newsfeed and now presents at the top the posts that it evaluates as the most interesting for the user. Their algorithm, sometimes referred to as EdgeRank, is not known to the public but uses the affinity between the poster and the user, the type of content posted and the time elapsed since its posting to determine to position it should occupy into the user's newsfeed. This new presentation suffers from the same drawbacks as the classical newsfeed, to few news are presented on the screen at any given time and it is nearly impossible for a user to locate the last activity of a given user without going to check her page.

Social visualization can be defined as the visualization of social data for social purposes [4]. In other words, social visualization uses information technology and focuses on people, groups and their conversational patterns, interactions and relationships with each other and with their community [4]. Social data may be collected from different sources such as online communities (e.g. IM logs, email archives, discussion threads, updates on social networks, etc.) and also the physical world (e.g. movement and location data captured by camera, GPS, mobile devices, etc.), and then be processed to generate visualization. Visualizations of these kinds of social data can be used for increasing awareness of one's social activities, motivating users to participate in social communities, and coordination. There are various social visualization approaches and techniques that have been proposed.

The Babble system [2] is one of the first approaches integrating the social visualization technology into an online chat room system. It was designed for a small- to medium- sized group, and to provide clues about the presence and activity of a person in the current conversation. Each person in the system is represented by a dot of different color. A gray circle in the center of the visualization represents the proxy of the current chat room. All users, who have already logged in to the system, but not in the current chat room, will be positioned outside the gray circle. The dots located inside the circle denote users who are in the current room. When people are active in the conversation, meaning they either "talk" (type) or "listen" (click and scroll), their dots move to the center of the circle, and then drift back out to the edge when they stop talking for 20 min. This way, everyone in the system is able to get the awareness about others' activities, which is good for the group coordination. This approach has also been used for some other online activities, such as online lectures and auctions.

Data portrait is another useful method for social visualization. For example, both PeopleGarden [8] and another floral representation [9] use this method to present social data. PeopleGarden is designed for online interaction environments such as web-based message boards, chat rooms, etc. In PeopleGarden, a flower metaphor, including magenta petal (for initial post) and blue petal (for response) has been used for each user in the system. Dots on the petal indicate the number of answers to this post. The height of the flower reflects how long the user has been in the system. Faded petals are used to indicate old posts. To visualize a group of users, all the flowers representing the users in this group can be drawn together on the canvas, which looks like a garden. In this way the current state of posts could be easily seen at a glance the visualization. For example, a healthier-looking garden with more bright flowers indicates a discussion group with more new posts. And the number of petals can also reflect how active the group is. Additionally, people may be motivated by the visualization to post more, and in this way get more petals for their flower.

IBlogVis [3] uses the digital footprints method to help a user find interesting articles when she is browsing blog archives. In IBlogVis, according to the time of posting, each blog entry is displayed as a point on the time line located in the middle of the page. A vertical line above each point (each blog entry) represents the length of each entry, and a second vertical line below each point represents the total length of comments this entry has collected. The circle's radius on the end of this line indicates the number of comments for each entry. A user can also click on an entry to view its content. This visualization application provides a rich overview of a blog. The history of social interaction (footprints) can help users identify potentially useful/interesting entries in a blog.

Motivational social visualization is an important category in social visualization. Its main purpose is using social visualization to motivate participants to contribute more or make higher quality contributions. For example, Comtella [7] is a peer-to-peer file-sharing community. The visualization in Comtella uses a metaphor of a night sky in which every user is represented by a star. The size of the star indicates a user's number of contributions (files shared in the Comtella community). A reddish coloured star represents a user who has shared more new files than the number of downloaded files from other users, and a blueish coloured star represents a user who downloaded more files than she has shared in the community. The big yellow star represents the "best user" who shares more than everyone else and has contributed new things to a community. Therefore, the visualization encourages social comparison among users to increase the diversity of resources in a community.

In general, the social visualization approaches discussed above have been applied to various online communities for different purposes. Facebook. as one of the most popular online communities, also has some visualization applications to help its users to explore their social data.

Facebook Social Graph[3] is one of the most popular social network visualization applications. The visualization in Facebook Social Graph shows how a user's friends connect to each other forming interconnected clusters (only the direct friends of a

[3] http://www.mihswat.com/labs/app/facebook-social-graph.

user are shown, but not the friends of friends), and it lets a user explore her social network by zooming in and out and panning the canvas. In this visualization, each node represents a user, and the line between nodes shows the relationship. All the nodes and connections are organized in a force layout. Placing the mouse pointer over a node can highlight all the mutual friends of the selected person and the current user. A pink circle emphasizes a social cluster (a group of mutually interconnected users) in the user's social network. This visualization can increase the user's awareness of the structure of her social network, which is more difficult to find out by exploring the list of friends of one's friends using Facebook's interface. Furthermore, being aware of social clusters can help the user predict how information flows among her friends.

Facebook Friend Wheel[4] takes a user's friends, then links and groups them together to form a colorful circular graph. Each node represents one friend, and a line connecting two nodes means that they are friends with each other. Nodes of different colors reflect different social clusters in the user's social network. All the nodes are organized in a wheel style, which can display the relationships more clearly. Similar to Facebook Social Graph, Facebook Friend Wheel only reveals the network of the user's direct friends, and provides the social map for getting social network awareness.

Facebook visualizer[5] is a tool to graphically discover the social network with a filter functionality including gender and relationship status. The major difference with the previous two Facebook visualization applications is that the application is interactive and allows the user to define filters to customize the display of a user's friends, so a specific group of friends can be hidden or shown to reduce the graph's complexity.

Nexus[6] calculates friend similarity by parsing profiles (through the Facebook API), and highlights links between friends who share the same interests and groups. This visualization in Nexus aims at classifying friends with same interests and groups, increasing the awareness of a user about the people in her social network.

Most of these third-parties' visualization tools focus on reflecting the social network structure. There is still no effective way to allow users to find some important posts that they may have missed, if they have not logged in for a couple of days, to find out whether a particular user has posted something recently, or who is generally active on Facebook and who is just a lurker.

This kind of awareness is useful not only for a user to build a trusted social network, but also for a community to understand its members' participation. Therefore, this paper proposes an interactive visualization approach that allows discovering the time patterns and the main current contributors, as well as the lurkers. It also allows users to browse their Facebook streams in an alternative way.

---

[4] T. Fletcher. Friend Wheel. http://thomas-fletcher.com/friendwheel/, 2009.

[5] http://vansande.org/facebook/visualiser/.

[6] http://nexus.ludios.net/.

3 Proposed Visualization

Facebook presents all the user's friends' social data in a stream, organized in a reverse-chronological order. Users log in periodically and browse the updates of their friends going back in time. However, this way of browsing has limitations:

1. It is often overwhelming to view the posts, especially if the user has not logged in for a long time, or if her friends have been very active during her offline period.
2. It is very easy to miss posts that could be potentially interesting, i.e. from the friends the user cares about. While Facebook provides the option to check the updates of a specific friend, it is not easy, since only a few friends are presented at a time on the screen, and to find a particular friend, a user has to search for him/her. This does not fit in the casual browsing pattern in which users normally explore Facebook, and the result from such a search would often be disappointing, since the friend in question may not have posted anything recently.
3. It is impossible to get an overall picture of who has posted updates recently, how recently, how many updates, and which of the friends have not been active. The focus of attention of a user falls naturally on the friends whose updates can be seen in the current stream, with perhaps one or two clicks back in history.

For these reasons, it is important to make users visually aware of:

1. Who has posted and at what time;
2. The number of posts, i.e. how active the user has been recently;
3. How popular the posts are, i.e. how many likes and comments were received by each post.

The goal of the proposed visualization, called Rings,[7] is to ensure an alternative way of browsing the stream of updates on Facebook, which allows the user to see which of her friends has been active recently and to check selectively the latest updates posted by her friends (instead of scrolling down through all the updates in the stream). This will reduce the cognitive overload of the user and will allow her to quickly check posts by particular friends, to be aware of (and possibly ignore) the most active users, and also to be aware of the users who are not posting and may be lurkers.

There are several requirements that have to be considered in the design. It is important to consider the scale of the visualization, i.e. how many friends it has to display. From the official statistics of Facebook, the average user has about 130 friends. Therefore, the visualization should be capable to arrange and display at least 200 users at the same time on the screen.

Additionally, the visualization should provide the option to easily locate a friend (since it could be challenging to identify a particular individual among 200 others) by providing a "search friend" option. A user should be able to see the most recent posts of each friend in such a way that it does not obstruct viewing the visualization as a whole. Jumping to the Facebook page of a friend when a user clicks on her profile

[7] http://rings.usask.ca.

picture or one of her posts, should also be supported, so that a user can interact with her friends on Facebook in the usual way, by posting comments to their friends' updates etc.

The design includes each individual user's representation in the visualization (for simplicity, it will be called "avatar"), visualization layout, functions, and application user interface. The avatar focuses on how to reflect the number of posts from a user during last 30 days in the visualization. How to arrange a large quantity of avatars in a neat and appealing way is a challenge that the visualization layout has to address. Rings' user interface and functions aim at providing an easy way for the user to navigate in the visualization and access the usual Facebook content through it.

Avatar Visualization. In Rings, each user is represented as a spiral (also called "screw"). The number of the posts in last 30 days is scaled into one of the six different levels of contribution (from level 0 to level 5). To visualize these levels, different sizes of spirals are applied to represent the six levels (see Fig. 1). Considering that it would be hard to distinguish six different sizes of spirals on a screen possibly crowded with many spirals representing hundreds of users' friends, it was decided to use an additional symbol variable—color, to differentiate the level of contribution of the user represented by an avatar. The color variable duplicates the meaning of the size variable, but it is easier to distinguish the different levels of contribution.

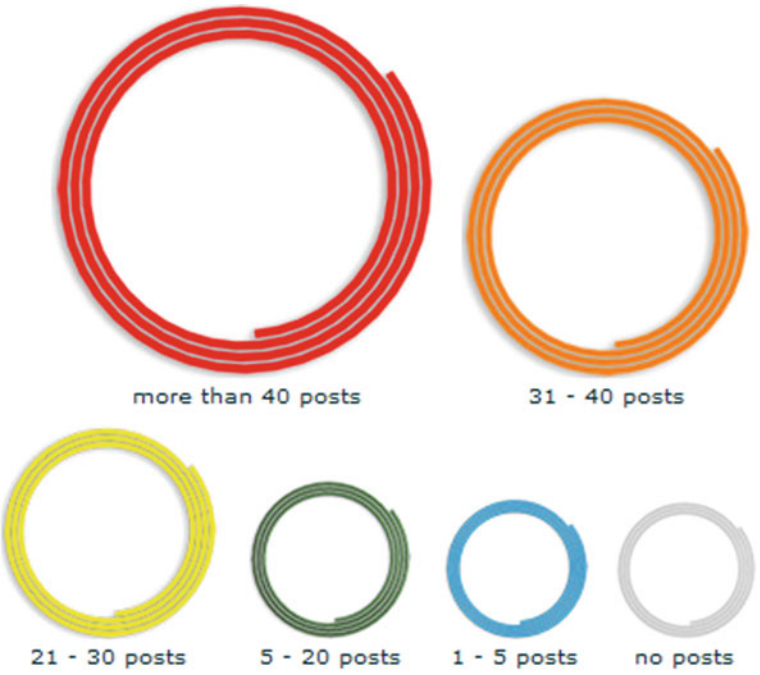


Fig. 1 Each level of quantity is indicated with a specific color and size

Yet, there is obviously a risk with overloading the representation of just one feature (the level of contribution), since it may lead to confusion in some users. For this purpose, a legend is provided, so that users can find at any time the meaning of the different sizes and colors of their friends' avatars. In addition to this, the related usability research shows that approximately 10 % of human males, along with a rare sprinkling of females, have some forms of color blindness.[8] Thus, the six colors are carefully chosen and tested under all the forms of color blindness on Colblindor (http://www.colblindor.com). In order to help users recognize their friends more easily, the profile picture and the username of each user on Facebook are displayed in the spiral, along with the number of posts the user has contributed during the last 30 days as shown on Fig. 2.

In order to see the posts made by one of her friends, the user only has to hover the mouse over her friend's avatar to see a detailled list of her friend last 30 days activity. To reflect how interesting/popular posts are, the numbers of likes and comments they receive are used. Obviously, the more likes and comments a post has, the more interesting it is. According to the total number of likes and comments, each post is classified into 5 different popularity levels displayed with different emphasis on the screen by means of different shades of gray. All the 5 levels are presented with 5 different gray colors [1]. For example, a post with many likes and comments is shown in solid black color, while a post with no likes or comments is shown in light-gray color. Additionally, to indicate the exact numbers of likes and comments, a bracket with two numbers is added at the very beginning of each post in the floating window if there are some likes and comments for this post. For instance, [L:4 C:3] means there are 4 likes and 3 comments on this post (see Fig. 3).

This strategy is also applied to the avatar visualization (spiral/screw) on the screen to provide awareness for the user to see at a glance which Facebook friends have some interesting/popular posts. As discussed in the last paragraph, each post is classified into one of the five different popularity levels according to the total number of likes and comments. Similarly, the avatar visualization is also classified into one of the five different levels according to the highest popularity level of posts that the user has got and five different opacity levels are used to present the five different popularity

Fig. 2 The profile picture and the username are displayed in the spiral

[8] A. Wade. Can you tell red from green? http://www.vischeck.com/info/wade.php, 2000.

Fig. 3 Opacity levels for posts

Fig. 4 Opacity levels for avatars

levels of that user as shown on Fig. 4. For example, if a user has a post at the most interesting level (in black color), her avatar visualization will be displayed in the highest opacity level of the corresponding color of the avatar (will look most solid). If all the posts of the user have no comments or likes at all, that avatar will be shown in the most transparent opacity level.

Layout. The number of friends varies drastically among Facebook users. For example, there are quite a few users with over 1,000 friends. Considering the acceptable loading time, the unavoidable timeouts of the Facebook API, and the resulting crowded screen, it is impossible to display all the friends of such a user on the screen at the same time. Therefore a restriction was introduced in the design on the number of friends that can be displayed in one screen. If a user has more than 200 friends on Facebook, they will be separated in groups of equal size $k < 200$. The user can select any of these groups to display. Then the visualization will only display these selected k friends after an acceptable loading time.

In order to represent how much time has elapsed since the latest post by a specific friend, the background layout was designed as a set of concentric rings, where the friends who have posted most recently are displayed in the center, and people who have posted long time ago will be shown at the periphery. There are several rings on the screen to indicate different time periods in the past. The rings, from the center to the periphery, show the last 3 h, last 12 h, last 24 h, last 3 days, last week, last 30 days, and no posts. Each avatar (spiral) representing a Facebook friend is placed on a specific ring according to the post-time of her latest post as represented on Fig. 5. For example, a user will show up at the very center in the visualization if she posted something in the last 3 h. If she stops posting anything from then on, her spiral will keep drifting to the periphery in the visualization over the next 30 days and will finally settle somewhere on the outmost ring.

Since research shows that humans naturally tend to focus their attention to the center of an image, the user's attention will focus on the most recently active users,

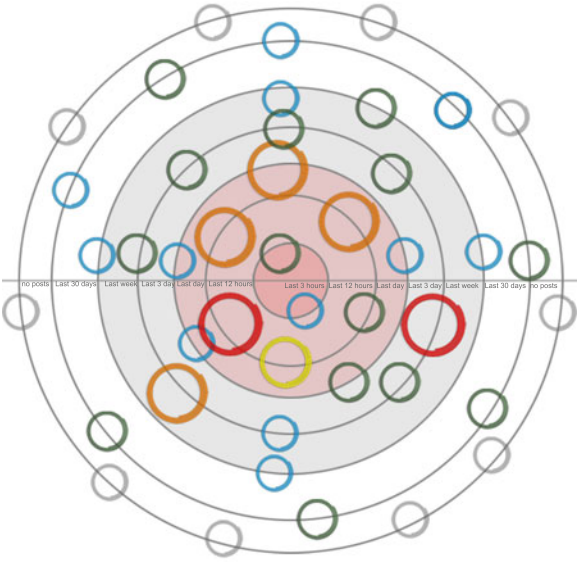


Fig. 5 Layout of the visualization

similar to the default display option in most streams (the most recent at the top). This design also naturally focuses the attention of the user to the center (the "Bull's Eye"), where the action is, and the most recent posts are. By exploring the visualization, the user will become aware of who among her friends were active during the last 30 days (avatars inside the outmost ring), who posted recently (avatars close to the center) and who has been active several days ago, who has stopped posting updates and has possibly become a lurker (avatars on the outmost ring with 0 posts), who posts regularly many updates and who tends to post only occasionally.

The concentric rings design allows for scalability, since the time periods represented with concentric circles grow "exponentially" (although not in the mathematical sense of the term). There will be fewer people who posted very recently and the space in the center is limited, while there will be many more people who have posted in the past, the more distant the past, the larger the ring and more space available to accommodate more avatars without being crowded.

Functions. All the functions available to the user in Rings are in the Tool box, which is made of three tabs as seen on Fig. 6, one for each function.

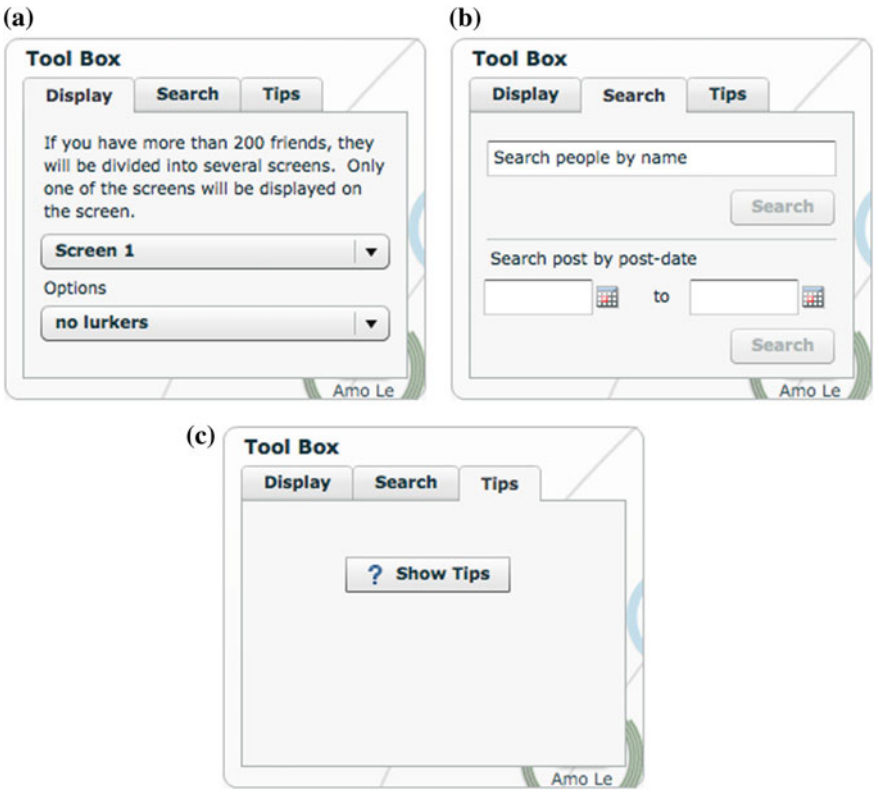


Fig. 6 Tool box presentation. **a** Display tab. **b** Search tab. **c** Tips tab

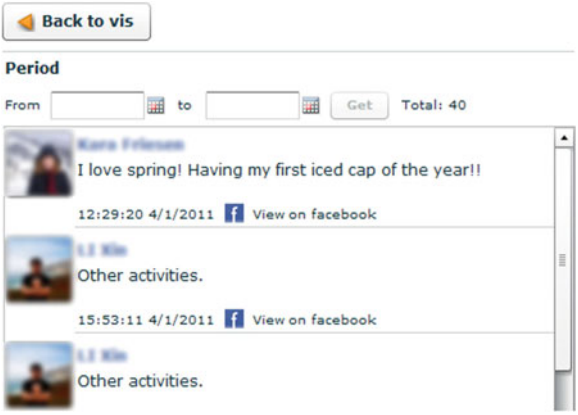


Fig. 7 Posts search results presentation

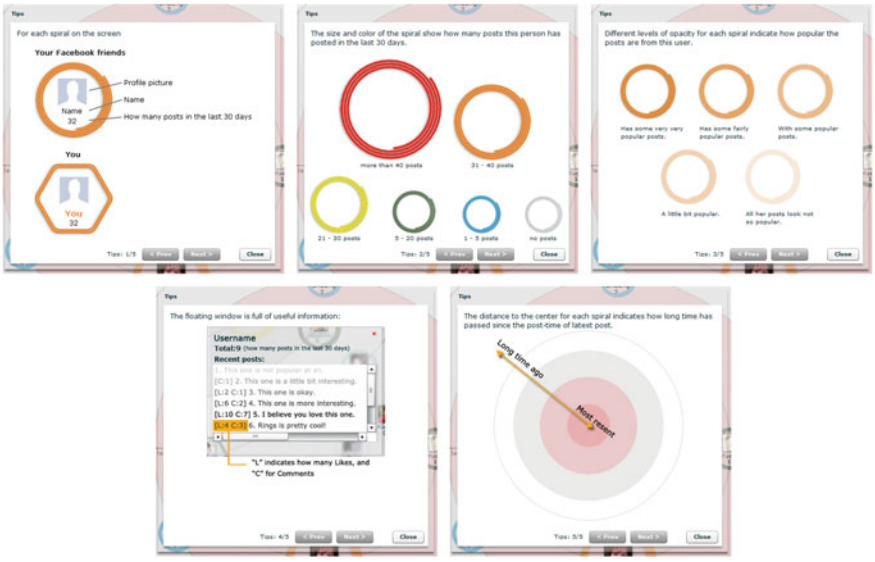


Fig. 8 Tips window

The display tab allows the user to navigate between the different groups of friends if she has more than 200. Also she can filter the layout by only displaying a specific group of friends or removing the lurkers (users without any activity in the last 30 days).

The search tab allows the user to highlight a friend in the layout to identify her position faster. The selected friend avatar will blink for some time to help the user find her. Also it is possible to search for activities posted in a certain timeframe. The activities that happened in the specified timelapse will appear in a list as shown on Fig. 7.

The tips tab allows the user to see Rings' tips as depicted on Fig. 8.

4 Evaluation

Rings was evaluated through two user studies. The first study was done in a lab under the observation of one of the authors. The second study was a continuous remote user study.

4.1 First User Study

This first study aims at testing the following three research questions.

1. Whether the information provided by the vizualisation in Rings allows an increased awareness of specific friends' Facebook activities (i.e. who is an active user, who is a lurker, etc.) or not?
2. Whether the information and functions provided by the vizualisation are useful for the user's daily Facebook browsing or not?
3. Whether the user interface is convenient and usable and the performance of Rings is good or not?

Sample. Testing the research questions mentioned above is not the only goal in the first study. Professional suggestions about the user experience and the visualization layout are also desired. Therefore, email invitations were sent to students of two research labs, the MADMUC Lab and the Human Computer Interaction Lab, at the Department of Computer Science, University of Saskatchewan. Additionally, two graduate students from other labs were willing to participate in the user study. Finally, there were eleven participants recruited, including six (6) students from the MADMUC Lab, three (3) students from the HCI Lab, one (1) from the Bioinfomatics Lab, and one (1) from the Agents Lab. All of them had strong computer background and advanced computer skills.

Because Rings is designed and implemented as a Facebook application, a diversity of participants in terms of Facebook experience was obtained to allow a better sense of how they use Facebook normally with their variety of Facebook experience levels. Two parameters were used to identify the Facebook experience level for each user: the number of Facebook friends they have, and the daily number of hours they usually spend on Facebook. Among all the participants in the first user study, more than half of the participants (6 out of 11) have 100–299 friends, followed by two (2) participants who have 50–99 friends and two (2) participants who have less than 50 friends. One (1) participant has more than 300 on Facebook (Fig. 9).

For the number of hours spend daily on Facebook, eight (8) participants browse Facebook less than one hour per day. One (1) participant spends 1–2 h, followed by two (2) participants who use Facebook more than 2 h daily (Fig. 10).

Procedure. All the participants were invited to a quiet and small office to avoid unwanted interruption. They were asked to fill a brief questionnaire about personal information, including gender, number of Facebook friends, daily Facebook browsing hours, and computer skill. After that, the researcher introduced some terms,

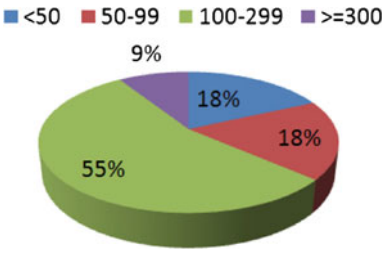


Fig. 9 The number of Facebook friends of study participants

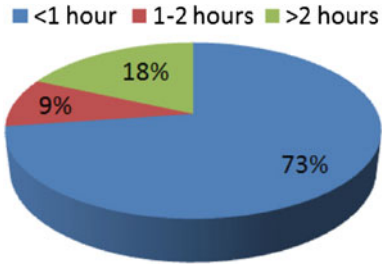


Fig. 10 The daily Facebook using hours of study participants

features (i.e. "lurker", friend-list feature on Facebook) that participants may not know, matters that needed attention, the flow of this study, and informed participants that the entire study would be captured with audio and screen recording.

During each study session, a participant needed to complete two sets of tasks: tasks to be completed on Facebook and tasks to be completed using Rings (Appendix A.1). The whole process of task completion was observed by one of the authors. To avoid differences in the amount of information that they had to tackle, which would have been unavoidable if the participants were to use their own Facebook accounts, all the participants were asked to use the same example Facebook account (it was a real account with 198 Facebook friends), for both sets of tasks (the Facebook task set and the Rings task set).

Among these two sets of tasks, there is one section, which comprises the seven core tasks, in the task set for Facebook. Three sections, the task about the login of Rings, tasks about the user interface, and the seven core tasks, were included in the task set for Rings (Fig. 11).

The goal of the seven core tasks was to evaluate and compare the user performance on each of these tasks using Facebook and Rings, along three parameters: time needed to complete the task, rate of correct response (task performance), and level of difficulty (a subjective user measure of how difficult the task is). The result of the comparison can be used to test the first research question. After completing these seven tasks using Facebook, the participants were asked to state how useful or related each task was (level of usefulness) to their daily Facebook browsing, which can answer the second research question.

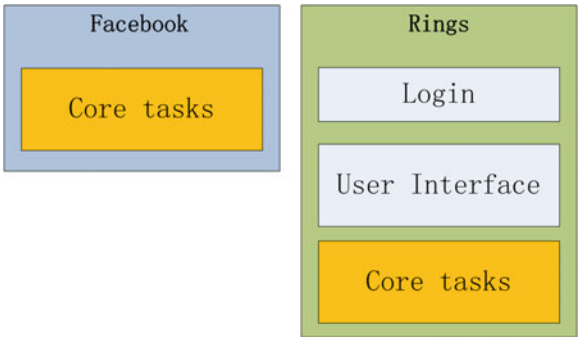


Fig. 11 The task sections in the two sets of tasks

One complication arises due to order effects (the order in which the participants perform the two sets of tasks), which can confound the accuracy of the results. There was one kind of order effect expected for the first user study: the experience, as well as the answers that a participant got from the first part of the experiment (using Facebook or Rings) might affect the result when she was working on the same task in the second part (using Rings or Facebook). For example, a participant was asked to find 3 users who had never posted anything in the past 30 days, which was anticipated to be hard to finish with the normal Facebook interface, but was much easier to do if the participant used Rings. If the participant first completed the set of tasks with Rings, she might remember the answer to each task, and take a shorter time to finish the same task using Facebook.

To deal with the order effects, a balanced within-subjects design was utilized in the first user study. All the participants were divided into two groups. The first group completed the 7 core tasks using Facebook first, and then finished the same tasks with Rings. For the other group, participants completed all the tasks in Rings first. In the task set for Rings, the sections about login and user interface were used to test the third research question (Appendix A.1, Tasks for Rings). Following and working on these tasks could also help participants to get familiar with the interface and fuctions provided by Rings.

After each session during the first study, a questionnaire related to the tasks was provided to collect general opinions about how easy it was to finish each part of tasks, comments, and suggestions. Some questions in the questionnaire were open ended, which enables participants to describe their own ideas or suggestions without any restriction. Additionally, the Questionnaire for User Interaction Satisfaction 7.0 was used to collect feedback about the overall user reactions, screen, terminology and system of information, and system capabilities.

A pilot study with two volunteers was conducted before the actual experiment to find shortcomings and issues of the designed study, as well as bugs of Rings. The bugs and issues collected from the pilot study were fixed before the formal study started.

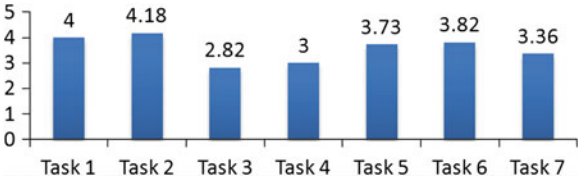


Fig. 12 The level of usefulness (means)

Results

Results for the core tasks

For the seven core tasks, the levels of usefulness for all the tasks were calculated and analyzed to test whether the information and awareness provided by the visualization in Rings are helpful or useful for daily Facebook browsing (the second research question). The level of difficulty, the elapsed time, and the correct rate were compared to test the first research question.

From Fig. 12 we can find that most (six) of the tasks were stated as neutral (level 3) or useful (higher than level 3) for daily Facebook browsing. Only one task, task 3 (the lurker awareness provided by Rings), was rated lower than 3 indicating that most of the participants thought the awareness of lurkers was not so useful for daily Facebook browsing.

As discussed earlier, one order effect was expected before the first user study: the experience, as well as the answers that a participant got from the first set of tasks (using Facebook or Rings) might have affected the result when she was working on the same task in the second part (using Rings or Facebook). According to the real observation of the first eight sessions of the first user study, the order effect did not appear among the first seven participants (including four participants who completed the seven core tasks using Facebook first, then used Rings, and three participants who used Rings first and then Facebook). However, that order effect was recognized during the eighth study session. That user finished all the seven tasks smoothly using Rings first, but then got trouble on a task using Facebook. Hence, she just wrote down the answer that she got using Rings in the previous section, without actually working on that task using Facebook. Therefore, all the following three participants were asked to use Facebook first then use Rings to complete the seven tasks.

Figure 13 presents the comparison of the level of difficulty (means) for completing each task using Facebook (blue bars) and Rings (red bars). The results clearly show that Rings provides an easier way for participants on searching specific task information.

The elapsed time (mean) and correct rate (mean) for each task with Facebook and Rings are compared to reflect the difference between two methods (Tables 1 and 2, Figs. 14 and 15). From the overview of the data shown in the tables and figures, one can see that the participants used less time to finish each task using Rings, and the

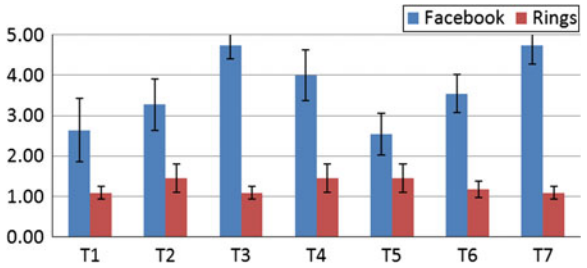


Fig. 13 The comparison of difficulty (means)

Table 1 The mean and standard deviation (STDEV) values of elapsed time and correct rate (accuracy) using Facebook and Rings—Group 1 (Facebook first, then Rings)

Task	Group1 (Facebook (FB) first, then Rings) 7 participants							
	Time with FB (s)		Time with Rings (s)		Accuracy with FB		Accuracy with Rings	
	Mean	STDEV	Mean	STDEV	Mean	STDEV	Mean	STDEV
T1	83.29	29.36	46.29	18.55	0.77	0.29	1.00	0.00
T2	135.57	45.03	78.14	21.26	0.77	0.27	1.00	0.00
T3	187.57	84.62	38.14	13.61	0.86	0.38	1.00	0.00
T4	294.00	167.03	80.43	23.11	0.86	0.15	0.94	0.15
T5	180.14	71.83	109.57	55.48	1.00	0.00	1.00	0.00
T6	109.29	30.10	52.57	16.08	0.89	0.13	1.00	0.00
T7	1,000.00	0.00	26.14	8.63	0.00	0.00	1.00	0.00

Table 2 The mean and standard deviation (STDEV) values of elapsed time and correct rate (accuracy) using Facebook and Rings—Group 2 (Rings first, then Facebook)

Task	Group2 (Rings first, then Facebook (FB)) 4 participants							
	Time with Rings (s)		Time with FB (s)		Accuracy with Rings		Accuracy with FB	
	Mean	STDEV	Mean	STDEV	Mean	STDEV	Mean	STDEV
T1	27.75	18.95	79.25	32.19	1.00	0.00	0.88	0.25
T2	98.75	33.88	187.25	25.70	0.98	0.04	0.73	0.27
T3	44.75	20.21	199.25	104.04	1.00	0.00	0.92	0.17
T4	103.00	68.88	234.00	210.67	0.85	0.19	0.80	0.28
T5	131.00	27.87	166.25	37.81	1.00	0.00	1.00	0.00
T6	52.75	10.34	135.25	111.31	0.94	0.13	0.94	0.13
T7	34.50	6.45	1,000.00	0.00	1.00	0.00	0.00	0.00

correct rates (accuracy) of the answers they found with Rings were higher than those found using the normal Facebook page for most of tasks. The participants got all the correct answers with both methods on task 6, but more time was used to complete that task. All the participants thought it was really difficult to get idea for the answer

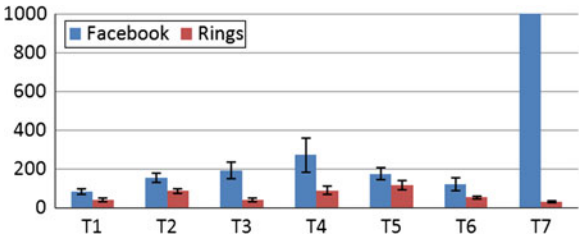


Fig. 14 The comparison of the elapsed time (means)

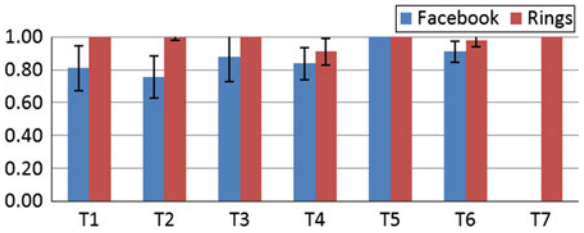


Fig. 15 The comparison of the correct rate (means)

of task 7, since there is no effective way for users to search posts by post-date on Facebook News Feed page. Therefore, an estimated time, 1,000 s, and a correct rate, 0, are assigned to all participants for this task.

Testing statistically if these data confirm the research questions can yield a more convincing results. Hence, the t-test method was applied on the elapsed time and correct rate for all the seven tasks. The t-test is one of the most commonly used procedures for hypotheses testing. There are 4 frequently used t-test methods: one-sample location test, paired two-sample for means, two-sample assuming equal variances, and two-sample assuming unequal variances [5]. In this user study, the 11 participants have varied backgrounds (computer skills, Facebook experience, familiarity with tasks), and the samples from participants using Facebook (and Rings) do not follow the same distribution, which means they are not comparable. However, the differences of elapsed time and correct rate between using Facebook and Rings are only caused by the difference between Facebook and Rings. Thus, the difference between the values of elapsed time and correct rate over the different tasks using Facebook and Rings are comparable and the paired two-sample for means method can be applied to test the research questions.

Before applying t-test, the samples of elapsed time and correct rate for each task using the two different methods (Facebook and Rings) from each of the participants should be aligned and listed, which can show the differences between two methods more clearly. For the elapsed time (Table 3), column Facebook in the table shows all the times elapsed using Facebook, and column Rings shows the times using Rings. Obviously, the difference in elapsed time between using Facebook and Rings for each task can be calculated by ($t_{Facebook}$—t_{Rings}) and is listed in column "Diff".

Table 3 Elapsed times and difference for each task (average values)

	Facebook	Rings	Diff
T1	81.82	39.55	42.27
T2	154.36	85.64	68.73
T3	191.82	40.55	151.27
T4	272.18	88.64	183.55
T5	175.09	117.36	57.73
T6	118.73	52.64	66.09
T7	1000	29.18	970.82

Table 4 Correct rates and difference for each task (average values)

	Facebook	Rings	Diff
T1	0.81	1	0.19
T2	0.75	0.99	0.24
T3	0.88	1	0.12
T4	0.84	0.91	0.07
T5	1	1	0
T6	0.91	0.98	0.07
T7	0	1	1

Similarly, the difference (c_{Rings}—$c_{Facebook}$) of correct rate for each task between using Facebook and Rings is shown in Table 4.

Two sub-hypotheses, H_{t0} and H_{t1}, are defined to test whether the participants completed the tasks in shorter time using Rings than that using the normal Facebook News Feed page:

$$H_{t0} : mean\_diff_{time} <= 0 \quad (1)$$

$$H_{t1} : mean\_diff_{time} > 0 \quad (2)$$

H_{t0} represents using Facebook is faster than using Rings to complete each task, while H_{t1}, indicates the opposite opinion, which means using Rings is faster than using Facebook. To test which hypothesis is correct is to calculate the critical value (t $\approx$ 6.05), and to compare t with t critical ($\approx$1.67, in the case of $\alpha \approx 0.05$). t is greater than t critical, which means H_{t0} should be rejected and H_{t1} should be accepted. In other words, using Rings to complete all the seven tasks is faster than using Facebook, and the result is significant, or not due to chance, with 95 % confidence (Table 5).

In the same way, whether completing tasks with Rings can get higher correct rate or not, can be tested by two hypotheses:

$$H_{c0} : mean\_diff_{crate} >= 0 \quad (3)$$

$$H_{c1} : mean\_diff_{crate} < 0 \quad (4)$$

Table 5 *t*-test result for elapsed time

	Elapsed time with FB	Elapsed time with Rings
Mean	285.44	64.25
Variance	95770.17	1690.5
Observations (samples)	77	77
Hypothesized mean difference	0	
df	76	
α	0.05	
t stat	6.05	
$P(T \leq t)$ one-tail	2.5E-08	
t critical one-tail	1.67	

Table 6 *t*-test result for correct rate

	Correct rate with FB	Correct rate with Rings
Mean	0.74	0.98
Variance	0.14	0.1
Observations (samples)	77	77
Hypothesized mean difference	0	
df	76	
α	0.05	
t stat	−5.60	
$P(T \leq t)$ one-tail	1.6E-07	
t critical one-tail	1.67	

Table 6 shows the t-test result on the parameter of correct rate. t can be calculated as about ∼5.60, and t critical is about 1.67. Thus, H_{c0} should be rejected, and H_{c1} should be accepted, which indicates using rings to complete tasks can gain higher correct rate.

Results of login

All participants thought it was easy (3 participants) or extremely easy (8 participants) to get connected with Facebook on the login screen (Fig. 16).

Figure 17 shows the user feedback about the speed of retrieving user data from Facebook on the login screen. The answer options for the level of speed were on a likert scale of 1 (slow) to 9 (fast). Seven (7) out of 11 (63.6 %) participants felt retrieving the stream of data about the 198 friends (that the Facebook account they were accessing had) from the Facebook server was fast (greater than or equal to 7). Two (2) participants rated the speed as 3, indicating that they were not so satisfied

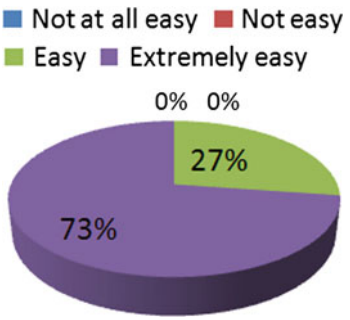


Fig. 16 The level of difficulty for login

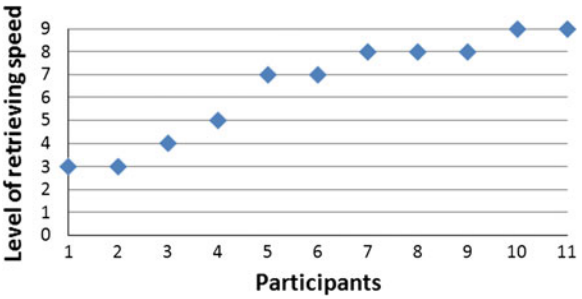


Fig. 17 The distribution of feedback by participants for level of retrieving speed (sorted by level of speed)

with the retrieving speed. Another 2 users chose the medium level (4 and 5) for the speed.

Evaluating the user interface

To evaluate the user interface, an initial list of 15 tasks (second part of Appendix A.1) involving specific functions of the user interface of Rings, was provided to all participants. The purpose of completing these tasks was to let users get familiar with the user interface, so that they could make a better use of Rings while working on the 7 core tasks. After finishing these 15 tasks, the questionnaire asked each participant how difficult it was to get a sense of user interface. There are four options for this question: not at all easy, not easy, easy, and extremely easy. Four (4) participants (36 %) felt that it was very easy to get sense of the user interface, and the other 7 participants thought that it was easy (Fig. 18).

Other results

Some overall questions about Rings were asked at the end of questionnaire. Figure 19 shows that 7 participants (64 %) stated that the information and functions provided by Rings are extremely useful while exploring Facebook. The remaining 4 participants considered that information and functions provided by Rings are useful.

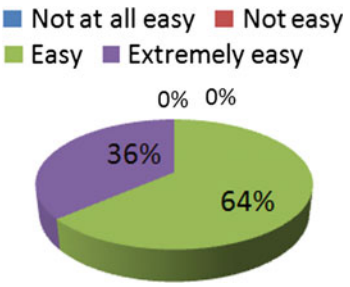


Fig. 18 The level of difficulty to get a sense of user interface

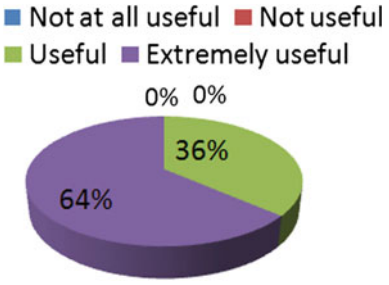


Fig. 19 The level of usefulness

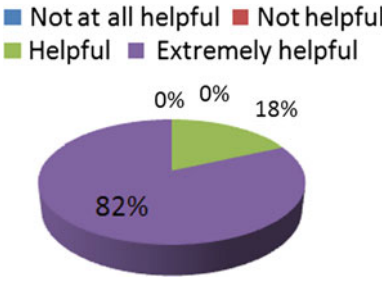


Fig. 20 The level of helpfulness

Furthermore, 9 participants stated that Rings did extremely well in allowing users to do things that could not be done easily with the normal Facebook pages, and 2 persons chose the option of Rings did well (Fig. 20).

Finally, when asked whether they would be interested in using Rings if was publicly available on Facebook, 6 participants were very interested, and 5 were interested in using it (Fig. 21).

Results of Questionnaire for User Interaction Satisfaction 7.0

The Questionnaire for User Interaction Satisfaction 7.0 was used to evaluate Rings from 4 different aspects: overall user reactions, screen, terminology and system of information, and system capabilities. The means and standard deviation values for each point in that questionnaire are shown in Table 7.

The results show that the participants found it was easier to complete the seven core tasks using Rings than using Facebook. The collected objective data (elapsed

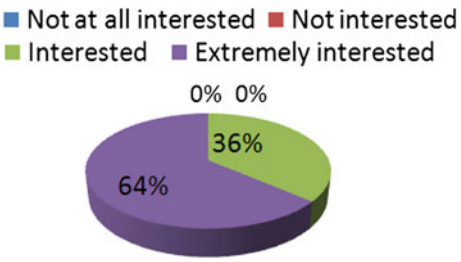


Fig. 21 The level of interest

Table 7 Mean and standard deviation values for each point in Questionnaire for User Interaction Satisfaction 7.0

			Mean	STDEV
Overall user reactions	Overall reactions to the application	Terrible (1) to Wonderful(9)	8.36	0.81
		Frustating (1) to Satisfying (9)	8.20	1.03
		Dull (1) to Stimulating (9)	8.36	1.21
		Difficult (1) to Easy (9)	8.36	0.67
		Inadequate power (1) to Adequate power (9)	7.73	1.27
Screen	Characters on the computer screen	Hard to read (1) to easy to read (9)	7.91	0.94
	The color scheme of the application	Ugly and hard to see (1) to Beautiful and easy to see (9)	8.55	0.69
Terminology and system information	Use of terminology throughout system	Inconsistent (1) to consistent (9)	8.45	0.93
	Messages which appear on screen	Confusing (1) to clear (9)	8.09	1.14
System capabilities	System speed	Too slow (1) to fast enough (9)	7.91	1.04
	The system is reliable	Never (1) to always (9)	8.45	0.52
	Provides useful feedback	Never (1) to always (9)	8.45	0.93
	Ease of operations depends on your level of experience	Difficult (1) to easy (9)	8.27	1.01

time and correct rate for each task) also proves that using Rings to finish the seven core tasks took a shorter time and resulted in a higher correct rate in comparison to using directly the Facebook page. The awareness provided by the visualization, and the functions provided in Rings were found useful by the participants for daily Facebook browsing. Moreover, the feedbacks about the user interface and user interaction show that it was easy to be understood and used, and the performance (speed of retrieval) was satisfactory. So the research questions can all be answered by the results in the first user study:

1. The information provided by the visualization in Rings allows an increased awareness of specific friends' Facebook activity (i.e. who are the active users, who are lurkers, etc.).
2. Most of the information and functions provided by the visualization are useful for the user's daily Facebook browsing.
3. The user interface is convenient and usable and the performance of Rings is good.

4.2 Second User Study

In this study, participants used Rings as they would normally use any other application. After each Rings' session, they were asked to fill a brief questionnaire.

Sample. The invitation message to the field study was sent via Author1 (224 friends) and Author3 (203 friends) Facebook wall pages, and it was also shared further by some of their friends who invited their own Facebook friends. Another invitation email was also sent to the students of the MADMUC Research Lab, University of Saskatchewan. Since the authors are in the area of computer science, there is a possible bias in the recruitment: some of the Facebook users who received the invitation message have a computer science background. The invitation offered two options: trying out Rings without participating in the study and using Ring as participant in the field study (after accepting a consent form). 21 users, including 7 participants from the MADMUC Lab, 4 from other labs in the Department of Computer Science, and ten unfamiliar Facebook users, agreed with the consent form to participate in this user study.

Procedure. All the participants were informed to use Rings for at least five days (one time each day, and at least 8 min for each time). After using Rings for 8 min, a button linking to the daily questionnaire appeared at the top-right corner of Rings, and kept blinking to attract user's attention. The user was required to click on that button, and then fill the brief questionnaire. The questionnaire contained 5 statements to which the users had to indicate their level of agreement or disagreement (varying on a scale of 5 levels from "strongly disagree" to "strongly agree"):

1. Rings gives me a good idea of who has posted recently on Facebook.
2. Rings makes it easy to find the most interesting stuff posted by my friends.
3. Rings makes me aware of who is very active on Facebook and who is merely lurking.
4. Rings' interface is easy to understand.
5. I intend to continue to use Rings as an interface to Facebook in the future.

After the participant had filled 5 times the short questionnaire (in different days, after using Rings for at least 8 min), a final questionnaire became available, containing 4 open-ended questions provided to solicit further user suggestions or comments.
Results. The study involved 21 users and continued for over 3 weeks. Each user was required to use Rings for at least 5 days before completing the study, but there was no

requirement that these days be consecutive. Some users stretched their participation for longer than 1 week and some completed the study in 5 days.

After the 3-week study, 5 participants completed all the five daily questionnaires (one questionnaire for each day), 1 participant finished four daily questionnaires, 2 filled two, and 6 filled one daily questionnaire. Totally, 47 daily questionnaires were submitted in the research database. One participant finished nine daily questionnaires, since there is no maximum limitation on the submitted daily questionnaires. A participant could continue with this study and fill the daily questionnaire after she completed all the five daily questionnaires. There are also eight participants who have never submitted any questionnaire.

The results obtained for the short daily questionnaire with 5 daily questions are presented in Table 8. The rationale for making the participants answer daily these 5 questions is that we expected to see their attitude change, as they gradually explore the functionalities of Rings and became familiar with the system. Therefore, the analysis regarding attitude change was conducted among the eight participants who submitted more than 2 daily questionnaires. Looking at Fig. 22, some attitude changes can be easily recognized according to the feedback from all five statements. Attitude change is most evident with respect to the second statement (Rings makes it easy to

Table 8 Results of the daily questionnaires

Question no.	Strongly disagree (%)	Disagree	Neutral (%)	Agree (%)	Strongly agree (%)
1	0	0	0	32	68
2	2	9	6	62	21
3	0	0	2	15	83
4	0	2	9	40	49
5	2	11	15	36	36

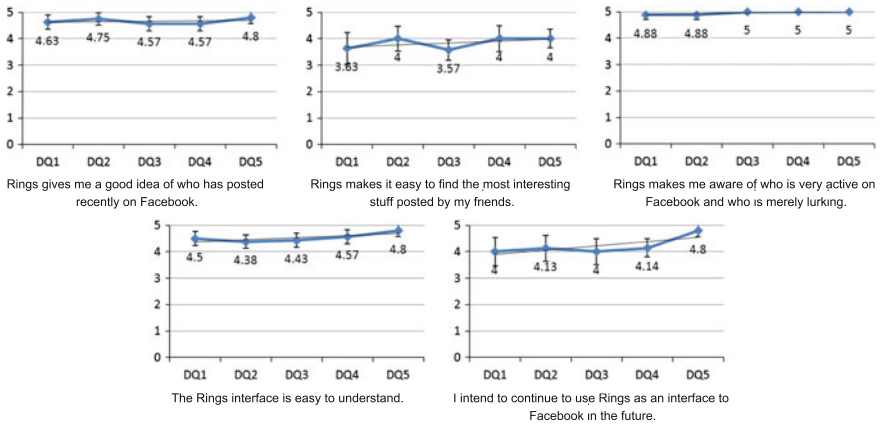


Fig. 22 Attitude changes on the five statements over the five daily questionnaires (DQ1, …, DQ5) (means)

find the most interesting stuff posted by my friends) due to lack of familiarity with Rings at the beginning of the study. However, after the third time of using Rings, the attitudes of the participants with respect to statement 2 changed towards "Agree" and "Strongly Agree." This pattern of attitude change can be also seen with respect to the other statements.

On the final questionnaire, four further questions about Rings were asked, which aimed at collecting further suggestions and comments from participants to improve Rings. These four questions asked the users about:

1. possible information they expected to find but which was not shown in Rings,
2. possible functionalities that were expected, but not provided,
3. anything that users felt inconvenient or uncomfortable,
4. other comments and suggestions.

Four out of five participants who finished all the five daily questionnaire filled the final questionnaire. The submitted feedback, revealed one limitation of the visualization, that it cannot provide information about interactions among the user's Facebook friends at a glance, since it is focused on the posting status. For example, it does not reveal that user A has commented/liked user B's post. While users can see that a particular post has generated interactions with other users (comments or likes) by the color and the header of the post in the floating window, users have to click on a specific post and then jump to the Facebook page to view the particular interactions with other users related to this post, which is not convenient.

Another limitation is that currently the information in the floating window is all textbased, and the user cannot visually scan for the types of things that they are interested in, as they are used to doing in the Facebook stream of updates. Customizing the avatars of friends was suggested by one participant as the functionality that she had expected but was not provided by Rings. Facing a lot of avatars on the screen makes it hard to locate the friends whom the user is interested in. So the user suggested providing a simple way to customize friends by clicking on the profile picture and to set the friend's spiral visible in the visualization or invisible. So that user can easily hide the friends she does not want to view by scanning the visualization.

There are still some usability issues remaining in Rings. First, the friends names are not shown properly in the visualization, which makes users uncomfortable. Currently, the user's full Facebook name is cut if the length of the name is longer than a certain limit. For example, "Tina Hang" will be displayed as "Tina H", since the limit of the name string is set as 6 characters. This display issue is more severe when displaying longer user name (e.g. "Andrew Verylonglastname"). Second, the search functionality is not implemented perfectly. Rings just scans all the usernames and picks the first one that contains the typed keywords, which can lead to wrong results when there are several matches. For example, if a user wants to search "Ian Jordan" among her friends, she types "Ian" as the search keyword. Then Rings provides another person named "Liankuan Bin" as the search result, because "Liankuan Bin" contains the keyword "ian", and coincidently "Liankuan Bin" is listed before "Ian Jordan" in the user array. Search recommendation is not provided in the current Rings

either. Finally, Rings cannot resize the visualization (i.e. zoom in or out) automatically according to the number of spirals on the screen to fit best the screen space. These issues need to be resolved in future versions of Rings.

With these two studies we evaluated Rings alog different dimensions. The first study shows that Rings is a practical, easy to use and awareness enhancing visualization tool that provides advantages over Facebook classical stream view. The second study shows that Rings has a very short period of adaptation and that users find it more efficient in order to gather information from their social network.

5 Conclusion

This paper introduces an intuitive and interactive visualization creating an increased awareness in the user about her social network on Facebook and allowing her to get insight about the level and pattern of posting activities of her friends. It provides an alternative way to browse Facebook's social data stream.

It is really important to provide social networks users with a greater awareness of their friends' participation. It helps them not to be overwhelmed by the huge quantity of "uninteresting" information and easily find the high quality information. Others visualization tools for social networks focus on increasing the user's awareness of the structure of her social network. Rings helps the user getting a better understanding of what is happening in her social network. The approach develop shows great promise. There are still several directions for the development of Rings.

First it will be really interesting to explore the angle in the visualization to position the avatars in proximity to each other, depending on different criteria, e.g. if they are friends with each other (in this way, it will create awareness of the structure of social network, something that is addressed already by other Facebook visualizations), or if they belong to the same organization, or share similar interests (addressed by other social graph visualizations, e.g. for LinkedIn). Various criteria for proximity can be used. In order to keep the main focus of the visualization on the time pattern of posts, the proximity would be secondary to the time pattern of posts, which is the main criterion for arranging the avatars on each ring.

Also Rings should take into account evidence of other user activities, such as liking, commenting, or just logging in or scrolling, to represent, rather than just number of updates and recency of updates. This would require enhancing the visual language to distinguish visually the different forms of activity. It would be an important extension since many online community users don't consider themselves lurkers, if they read, comment or rate [6]. However, this extension has to wait, as currently the Facebook API does not provide data on these activities for users.

Finally, applying a similar visualization to other social network sites, such as Twitter or LinkedIn and create a mash-up for all the users' social network sites is a natural extension of this work.

Acknowledgments This work has been supported by Natural Sciences and Engineering Research Council (NSERC) through the Discovery Grant program and the Discovery Accelerator Supplement Program.

A.1 List of Tasks

*Term "Active" is defined as "Post Frequently (more than 15 posts in last 30 days)" in this research.

Tasks for Facebook

1. Please try to tell all the users who have posted in last 3 h
2. Please try to tell all the users who have posted in last 24 h
3. Please try to tell 3 users who have never posted anything in the past 30 days
4. Please try to tell 5 users whom you think they are active users, and what are the post-dates of their latest posts
5. Please try to search Wendy Zhao, Lilia Li, and Jingyang Peng, count how many posts have been posted in last 30 days, and how long has passed since her last post for each person
6. Please only show the friend list named "U of S" on Facebook News Feed page, then tell 4 active users among them.
7. Please try to find all the posts between April 14, 2011 and April 15, 2011

Tasks for Rings

Login

1. Please connect to Facebook account using Rings

User interface

1. Please point out the control panel (comprises the "Tool Box", update timer, and scale slider) on the top-left corner, and visualization part (each friend is represented as a screw) in Rings
2. Please tell when Rings is going to check updates next round
3. Please point out the "Scale" slider, and scale the visualization
4. Please hide the "Tool Box", then show it again
5. Please tell the information contained in each screw
6. Please move mouse pointer on a screw, and tell the information inside the floating window
7. Please click on a post in the floating window to jump to that post on Facebook. Then back to Rings
8. Please click on a friend's profile picture to jump to his/her profile page on Facebook from Rings. Then back to Rings
9. Please go to "Display" tab in "Tool Box", and determine all the friends will be divided into several groups if there are more than 200 friends on Facebook. And only one of the groups can be displayed on the screen

10. Please hide all the lurkers with the "No lurkers" option under the "Display" tab
11. Please display the visualization with the "U of S" (friend list) option, then choose the "All" option
12. Please go to "Search" tab in "Tool Box", and search "shey", then find where is the result
13. Please search posts by post-time
14. Please go to "Tips" tab in "Tool Box", and click on the "Show Tips" button, then determine the meaning of different screw colors and sizes. Close the tip window when you are done
15. Please determine what the layout of the users' screws represents

Visualization

1. Please try to tell all the users who have posted in last 3 h
2. Please try to tell all the users who have posted in last 24 h
3. Please try to tell 3 users who have never posted anything in the past 30 days
4. Please try to tell 5 users whom you think they are active users, and what are the post-dates of their latest posts
5. Please try to search Wendy Zhao, Lilia Li, and Jingyang Peng, count how many posts have been posted in last 30 days, and how long has passed since her last post for each person
6. Please choose the "U of S" option under the "Display" tab, then tell 4 active users among them
7. Please search all the posts between April 14, 2011 and April 15, 2011, and tell the number of those posts you got

References

1. Alexander J, Cockburn A, Fitchett S, Gutwin C, Greenberg S (2009) Revisiting read wear: analysis, design, and evaluation of a footprints scrollbar. In: Proceedings of the 27th international conference on human factors in computing systems. ACM, pp 1665–1674
2. Erickson T, Kellogg W (2000) Social translucence: an approach to designing systems that support social processes. ACM Trans Comput-Hum Interact (TOCHI) 7(1):59–83
3. Indratmo J, Vassileva J (2008) iblogvis: an interactive blog visualization tool
4. Karahalios K, Viégas F (2006) Social visualization: exploring text, audio, and video interaction. In: CHI'06 extended abstracts on human factors in computing systems. ACM, pp 1667–1670
5. Montgomery DC, Runger GC, Hubele NF (2009) Engineering statistics. Wiley, New York
6. Nonnecke B, Preece J (2000) Lurker demographics: counting the silent. In: Proceedings of the SIGCHI conference on human factors in computing systems. ACM, pp 73–80
7. Sun L, Vassileva J (2006) Social visualization encouraging participation in online communities. Groupware: design, implementation, and use, pp 349–363
8. Xiong R, Donath J (1999) Peoplegarden: creating data portraits for users. In: Proceedings of the 12th annual ACM symposium on user interface software and technology. ACM, pp 37–44
9. Zhu B, Chen H (2008) Communication-garden system: visualizing a computer-mediated communication process. Decis Support Syst 45(4):778–794

Friends and Circles—A Design Study for Contact Management in Egocentric Online Social Networks

Bo Gao and Bettina Berendt

Abstract Users of Online Social Networks (OSN) may share private information with the "wrong" friends. To help users choose their audience better, we first designed a tool for the exploratory visualization of friend groupings. These groups ("circles") are formed by a hierarchical modularity-based algorithm for community detection (MOD). We then conducted a user study that showed that the hierarchical modularity-based circles were more helpful for users than Facebook's "smart lists" for making visibility decisions in online posting. We then compared MOD with another state-of-the-art community detection model, Generative Model for Friendships (GMF), finding that MOD circles were more similar to ground-truth groupings than those created by GMF. Extending MOD to produce overlapping circles further improved the results. Informed by our user study and research on social groups and information visualization theories in general, we developed an interactive friend-exploration/grouping web application for Facebook users.

Keywords Visualization · Community detection · Online social networks · Privacy · Web application · Design study · Contact management · Social grouping

1 Introduction

An Online Social Network (OSN) today can hold hundreds of millions of users. Two years ago (2012), Facebook (www.facebook.com) has reached its "one billion users" mark [64]. Behavioural and sometimes very personal information of OSN users is uploaded and shared online daily, in large quantity and tremendous detail. While the availability of these data enables us to understand more about our societies, it also

B. Gao (✉) · B. Berendt
Department of Computer Science, KU Leuven, Celestijnenlaan 200A,
3001 Heverlee, Belgium
e-mail: bo.gao@cs.kuleuven.be

B. Berendt
e-mail: bettina.berendt@cs.kuleuven.be

J. Kawash (ed.), *Online Social Media Analysis and Visualization*,
Lecture Notes in Social Networks, DOI 10.1007/978-3-319-13590-8_7

challenges us in effectively and efficiently processing large amount of information, and managing our online personal content.

In the age of online social networking, "even as bloggers and networkers delve into their private experience, they communicate with their fellow humans in a shared festival of the self" [9]. Such phenomenon has raised concerns about privacy. For example, it was found that OSN users had demonstrated high privacy concerns while revealing great amounts of personal information [1]. It also has been quantitatively demonstrated that users' perceptions of audience size do not match reality, since not enough feedback is provided [7]. Boyd showed that collapsed or ambiguous online contexts could lead to undesired disclosure of personal information [10]. Gürses extends the notion of privacy from confidentiality to access control and practice. The extended notion encompasses the solution space in which OSN users are empowered to re-negotiate the boundaries of information dissemination and construct their online identity based on a transparent system [30].

In light of the recent privacy research, we become interested in the tools that can help OSN users gain insights into their own social networks, explore to reveal hidden patterns and control the flow of personal data shared with online friends. As previous studies have suggested [26, 37, 48], in order to manage personal information flow, it is important for users to categorize their online friends into groups, categories, circles, lists or communities,[1] so that the user can post towards clearly specified audience. By "post", we mean the user's action of uploading or sharing digital information in OSN. We will also be using the term Egocentric Online Social Network (EOSN) to refer to a sub-network in an OSN, with the nodes representing people and the (directed or undirected) edges representing certain relationships among them. The network is centered on one user (as the ego), whose friends (as the alters) are directly linked to this user via edges. Edges usually also form among the friends.

As reported in 2011, the median number of friends of a Facebook user was 100 [57]. This number became 229 in 2013. For teens and people in their 20s, it was 400 or more [62]. To make sense of the increasingly complex online social networking data and manage online contacts, a user needs to deploy more sophisticated tactics (categorizing friends under different situations) than simple browsing and memorising. We started looking into visualization approaches to address this issue, as human visual system is highly parallel and pre-attentively sensitive to variations in visual stimuli, such as color, shape, positions, etc. [52]. With a carefully designed interactive visualization system, the user should be able to gain an overview of her network, explore the network to find novel patterns and easily construct groups of friends for different posting purposes.

The contributions of this chapter are: First, we document a user study, which shows that, compared with Facebook smart lists, the hierarchical modularity-based circles (used in an exploratory fashion) are more supportive for users to make

[1] We use these words interchangeably throughout the paper. The words "group" and "category" are used more generically, "list" is often used in the context of Facebook and Twitter (www.twitter.com). We use "circle" more often in the context of Google+ (www.plus.google.com) and visualization. The word "community" is usually used in the context of community detection algorithms.

privacy-related visibility decisions in online posting. Second, we design a new form of interactive visualization to visualize hierarchically grouped items. The items can be individual friends a user has in her online social network. Third, we test two community detection algorithms on three egocentric social network datasets. The results show that the ground-truth circles coincide more with the modularity-based circles. Fourth, we extend the modularity-based algorithm to accommodate the overlapping nature of online social circles. The experimental results show that this approach is indeed better than the original modularity-based algorithm. Fifth, we develop an interactive friend-exploration/grouping web application for Facebook users to explore their online social networks and create their customized friend-lists.

The structure of this chapter is as follows: Sect. 2 covers a set of existing tools for EOSN analysis and friend grouping. In Sect. 3, we analyze users' requirements, motivate and detail a new design of interactive visualization, named CircleTree. We then use it as the common interface to conduct a user study. The study compares users' behaviour in privacy decision-making based on two different friend-grouping mechanisms. In Sect. 4, we examine two community detection algorithms and discuss the nature of friend grouping in EOSN. We then propose an extended version of the modularity-based community detection algorithm to generate overlapping friend groups. In Sect. 5, with the introduction of the tool named FreeBu, we propose alternative views to supplement the earlier CircleTree visualization. We then identify the improvement points for the friend-exploration/grouping tool design. In Sect. 6, we conclude with a summary and an outlook on future work.

2 Related Work

We are interested in the tools that enable users to gain insights into their EOSN and/or construct friend groups. We describe a selective set of existing tools in Sect. 2.1, and discuss their relationships with our contributions in Sect. 2.2.

2.1 Existing Tools

PViz [38] is a tool that helps Facebook users understand their privacy settings. The tool is compared with existing policy comprehension tools on Facebook, namely Audience View and Custom Settings. It was shown that PViz was more effective for the users in comprehending privacy settings. Privacy Wizard [19] is a tool that can automatically predict the privacy preferences for a Facebook user based on her previous privacy-setting input, the result is also encouraging. Both tools employ Newman's Modularity-based community detection algorithm [46] to derive friend groups, which are then used for visualization (PViz) and prediction (Privacy Wizard) respectively.

NodeXL [53] is a general-purpose plugin that allows users to draw graphs by using a Microsoft Excel template. It implements various graph clustering algorithms, including modularity-based ones. It supports social network analysis, users can visualize their Facebook and Twitter graph data via an importer interface. The tool uses the Group-In-a-Box (GIB) feature [49] to help users delineate the clustering structure of the imported graphs. More specifically, the visual clusters are firstly formed in a graph layout. They are further constrained by being placed inside boxes whose sizes depend on the respective numbers of nodes. These boxes are then arranged by the squarified treemap algorithm [11]. The GIB layout is also used for multivariable grouping of the nodes based on their attributes. Some other general-purpose network-analysis tools are potentially useful for OSN users as well, such as Gephi [5], Cytoscape [51] and Tulip [3].

There also exist many small web applications that visualize OSN users' network data and allow simple interactions for exploration. Here we give two representative examples. Social Graph[2] is a Facebook application that shows a force-directed layout of the user's friend graph. The nodes are colored according to the detected communities based on Modularity [46]. The user can click on an individual friend (i.e. a node) to see the friend's profile photo along with three statistical numbers: the number of mutual friends she shares with that friend, the clustering coefficient of the friend and the clustering coefficient of the corresponding community. The user can further explore the graph layout by selecting a specific community from a drop-down list, so that only the members from that community are repositioned and displayed on the screen. Each community is labeled with the name of the friend in that community who has the highest clustering coefficient. The other example is InMaps.[3] It is a web application similar to Social Graph. It visualizes the user's network on LinkedIn (www.linkedin.com) with rather similar force-directed layout and modularity-based communities as well. Through InMaps, a LinkedIn user can zoom and pan to explore the map. The name labels of the friends are simultaneously brought to display upon zooming-in. We can also see that the nodes and labels are mapped with care to avoid overlapping, which makes the visualization more readable than Social Graph.

Personal Analytics for Facebook,[4] as part of the Wolfram Alpha knowledge Engine (www.wolframalpha.com), is a state-of-the-art visual and textual analytic web application designed for Facebook users. It offers a wide range of analytics, including various friends' demographic reports, summaries of the user's logging-in, posting and sharing activities, etc. Another merit of this tool is that each analytic segment can be downloaded in different formats for other uses, such as spread sheet, image and vector graph.

Furthermore, current Social Networking Sites (SNS) provide mechanisms for users to create their own friend groups, such as the lists in Facebook and Twitter, the circles in Google+ and the groups in Weibo (www.weibo.com). But by large, users have to manually group friends, which tends to become unmanageable. We know

[2] https://apps.facebook.com/socialgraph_fr_yl [Accessed on Nov 30, 2013].

[3] http://inmaps.linkedinlabs.com/network [Accessed on Nov 30, 2013].

[4] http://www.wolframalpha.com/input/?i=facebook+report [Accessed on Nov 30, 2013].

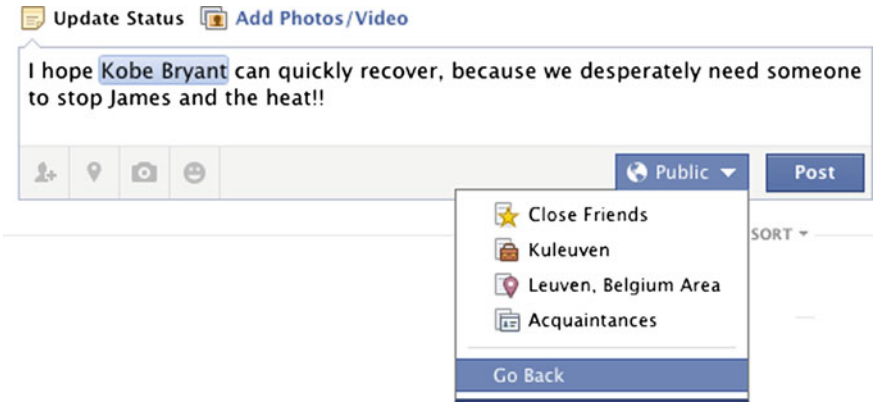


Fig. 1 A Facebook user can conveniently limit the visibility of her status by choosing one of the four lists, of which *Close Friends* and *Acquaintances* are the lists that the user manually defines, and *Kuleuven* and *Leuven, Belgium Area* are the automatically generated smart lists, based on the user's work and current city

one exception—Facebook "smart lists"—that can automatically generate friend lists. Facebook smart lists[5] provide users with an automatic grouping solution. The lists are generated based on the information about the user's education, work and current city. For example, if the user indicates Leuven as her current city, she will have a list with all of her friends who also indicate Leuven as their current city. The user can directly determine the audience of her posts by choosing one of the lists, including the smart lists. Figure 1 gives an example for status update.

2.2 The Tools' Relations to Our Contributions

Informed by PViz and Privacy Wizard, we find that one feature of Facebook's privacy control mechanism yet to be examined is the smart lists. In Sect. 3, we take the smart lists as baseline and investigate the roles that community detection algorithms and interactive visualization play in users' privacy decision-making process. We also note that PViz is for privacy-setting comprehension, Privacy Wizard is for privacy-setting configuration, both tools do not serve for the purpose of helping users create their own friend groups, e.g. Facebook friend lists. We built the interactive friend-exploration/grouping tool (as detailed in Sect. 5) to facilitate this activity.

The GIB feature in NodeXL currently does not support hierarchical graph clustering and exploration. A hierarchy is difficult to visualize and interact with, because the semantics from different layers may compromise the readability of a set of visual clusters, especially when the leaf nodes are of main interest (e.g. the

[5] https://www.facebook.com/help/204604196335128/ [Accessed on Nov 30, 2013].

user' friends). In Sect. 3, we introduce a hierarchical exploratory visualization design. This design displays the grouping structure of the user's EOSN and maintains the emphasis on the leaf nodes of a hierarchy.

Unlike Gephi, Cytoscape and Tulip, NodeXL offers its users connectivity to their Facebook accounts, so that they can easily analyze their ego-networks. Besides such connectivity, our tool also provides its users with a series of list-creation interfaces. The user-created friend lists can be submitted to their Facebook accounts. Moreover, NodeXL, Gephi, Cytoscape and Tulip are desktop applications/plugins that require installation. For NodeXL, the installation is conditioned on having Microsoft Excel in advance. Our tool is an online application that is easily accessible by a JavaScript-enabled browser. To facilitate user-defined friend-grouping, in Sect. 5, we elaborate the ways in which we improve the existing graph-layout visualizations, such as Social Graph and InMaps.

In Personal Analytics for Facebook, we find that the visualizations are fairly static as it follows the interaction syntax of a regular web page—that supports up/down scrolling and hyperlink clicking, but without zooming, panning and animation. Thus the action of inspecting individual objects in an overview visualization, e.g. foraging through the graph clusters, become problematic if the user has many friends. We consider our interactive visualization design to be complementary with respect to this.

Furthermore, we notice that all the aforementioned tools in Sect. 2.1 use or include modularity-based communities to approximate the user's social groups. Modularity maximization encourages mutually connected nodes to be put into the same community. While the broad adoption of this method is partly due to its popularity and software availability, another contributing factor seems to be that, among many other community detection methods, it produces the communities that best match the communities a user has in mind. Various studies have demonstrated useful applications of modularity-based community detection algorithms for social network analysis [35, 46]. We will also be using this method in our user study (Sect. 3). In Sect. 4, we examine this method in more detail and demonstrate its usefulness for EOSN friend-grouping. We also propose an extension of the modularity-based algorithm and show that it has a better performance.

3 A User Study on Circles for Visibility Decisions

This section consists of three parts: First, in relation to our user study, we discuss OSN users' need for friend-grouping tools (Sect. 3.1) and why we further choose to use hierarchical grouping (Sect. 3.2). Second, we examine the related work on visualizations for hierarchies (Sect. 3.3) and describe the CircleTree visualization that we have developed (Sect. 3.4). This visualization is then used in our user study. Third, we describe the participants, tasks and results of the user study (Sect. 3.5).

3.1 The Need for Grouping Tools

As discussed in Sects. 1 and 2, we know that the increasingly large amount of data produced by our online social networking activities has made it difficult for us to manage our personal information flow. Sharing certain information with the wrong people can cause awkwardness, embarrassment or even severe damage on the user. Therefore a tool is needed to inform OSN users and facilitate their privacy decision-making. More specifically, the user should be able to effectively determine which piece of her personal information is visible to which friend(s). But it would be a daunting task if the user goes through each individual online friend that she has one by one, and considers that friend's unique constellations of attributes and proclivities in order to make such a decision. In reality, informed by the research in social cognition [36], we know that people "prefer to construe others on the basis of the social categories to which they belong, categories for which a wealth of related material is believed to reside in long-term memory". Because of the limitations in human cognition and the challenges presented by a vast stimulus world (in our case—the online social networking environment, intertwined with the offline social life), a person naturally employs categorical thinking in order to simplify and structure the people she befriends [2, 36]. This description further provides support for the necessity of friend grouping [26, 37, 48].

We will be using the term Visibility Decision to refer to a user's binary decision on whether a post is visible to an individual friend in her EOSN. A post can be anything that a user uploads or shares in an OSN, e.g. a status update, a (re)tweet, a photo, a comment or an article shared, etc. Friend grouping can facilitate users' visibility decisions. The user decides the visibility of a post directly based on friend groups rather than individuals. In other words, when the user sees a group, assuming her previous familiarity with the group, she can skip the serial browsing that examines individual friends in this group, and determine the visibilities of the post towards those friends on a group level. There are two exceptions in which the user does not directly deploy the groups in her visibility decisions. *First*, certain posts are too privacy sensitive, i.e. it has become a complete regret, or not sensitive at all, in both cases, a binary decision becomes unary, and all user's friends are considered as one group. *Second*, when the number of friends who are or are not supposed to see a post (e.g. one, two or three) is significantly smaller than the number of friend groups shown to the user, then checking the groups requires more effort than just doing a standard search, e.g. typing friend names in a search box. Thus it is no longer necessary to use groups. However, we shouldn't completely disregard friend grouping in such situation, because it can raise the user's awareness about her friends, which can be useful for other aspects of life or later visibility-decision-making. Moreover, if shown appropriately, the friend grouping can help the user spot "unexpected" or "surprising" friends, which then becomes useful for the user to make visibility decisions.

3.2 Why We Use Hierarchical Grouping

In our user study (Sect. 3.5), we compare the two ways of detecting communities for a Facebook user—Facebook Smart Lists (FSL) and hierarchical modularity-based communities—in terms of their usefulness in facilitating the user's visibility decisions for posting. The original modularity-based community detection algorithm (MOD) takes the user's friend graph as input and produces non-overlapping, flat communities. There is a subgraph corresponding to each detected community of nodes. MOD is then applied to each subgraph, deriving sub-communities. We adapt MOD into a hierarchical one, abbreviated as HMOD. We choose to use HMOD for three reasons: (1) Modularity-based methods are known to have a "resolution limit" problem [22]. It is most likely that, for a community with $\sqrt{m}$ (m is the total number of edges) or less nodes, its sub-communities cannot be discovered. This implies that modularity optimization can miss the substructures of a network. (2) It is well known that people organize semantic concepts hierarchically in memory [13]. The reason for this is because storing generalized information with superset nodes is more economical for humans. Hierarchy is necessary in the navigation for the retrieval of more detailed information. (3) Another incentive that we use HMOD is based on the aforementioned Categorical Thinking, as iterative categorization (i.e. grouping) may be required from the user to make sense of the her friends if the number of friends is simply very large.

3.3 Related Work on Visualizations for Hierarchies

To examine the difference between two friend grouping strategies, there needs to be one common User Interface (UI). Given that a Facebook user usually has hundreds of friends, naively using "pen and paper" to elicit the visibility decisions from the participants may weary them. Bearing this in mind, we decide to let the participants operate on a computer-based UI. For users making visibility decisions with such an interface, we need two basic functions: *First*, browsing is applicable at both group and individual levels. *Second*, making a decision is applicable at both group and individual levels. Various existing works have paved the way for visualizing hierarchical grouping structure. We do not intend to provide a comprehensive review in this subsection. Instead, we give a qualitative treatment to four representative types of visualizations and motivate our design choices. We refer to the two dimensional area on the computer screen where a visualization is rendered as the canvas.

- *Node-Link Diagram* The traditional Node-link Diagrams use shapes (rectangles, circles, etc.) to represent nodes and lines to represent links. The direction from the root of the tree to the leaves is either vertical or horizontal. The nodes (intermediate or the leaves) at the same level need to be aligned at the same vertical or horizontal line. Hence only one-dimensional space is utilized to visualize each level. As shown in Fig. 2a, this space can be easily exhausted, especially at the leaf level.

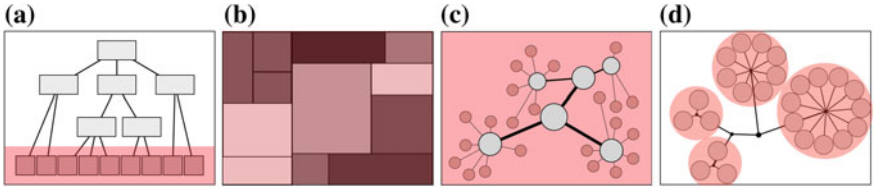


Fig. 2 Four types of representations for visualizing a hierarchical grouping structure, the potential area that can be used to draw leaf nodes is overlaid with *red* color. **a** Node-Link diagram. **b** Treemap. **c** Space filling tree. **d** Bubble tree

When the leaves are squeezed to be aligned and fit into the canvas, they easily become too small for the user to interact with, and the grouping structure is no longer clear at the leaf level. Improvements have been made using coloring and merging to reduce the number of branches and/or leaves to draw (e.g. Colored trees [50]). However, they leverage the continuous values of leaves, so that the colors correspond to different average values, giving a sense of numerical ordering. In our case, either the friends or the groups are discrete, which the user needs to differentiate to make a visibility decision. We also note that using the color visual channel to differentiate discrete variables (e.g. Stacked Tree [8]) is problematic, as there are very limited choices for visually distinct colors [28, 31].

- *TreeMap* Grid-based (or matrix-based) visualizations utilize the canvas space more efficiently, as shown in Fig. 2b. A typical grid-based layout is treemap [33]. It visualizes hierarchical data by nested rectangles. Many techniques have been proposed to make treemaps more structurally perceivable by humans. For example, shaded colors can bring a sense of ordering to the treemap nodes [54], gradient colors can demarcate different clusters in a treemap (cushion treemap) [58], the aspect ratio of the nodes can be adjusted to improve their readability (squarified treemap) [11]. Compared with node-link diagrams, treemap is more readable for various large-graph-related tasks, but path finding is consistently in favor of node-link diagrams [27]. More importantly, the user cannot conveniently select all the friends in a (sub-)group at once to make a visibility decision in treemaps.
- *Space-Filling Tree* Given the limitations in node-link diagrams and treemaps, hybrid visualizations have been proposed. The space-filling tree [47] is a typical example, as shown in Fig. 2c. It spreads the nodes and leaves across the whole canvas. To give a sense of structure, the sizes of the nodes decrease with ascending levels of the tree, the child nodes are mapped in proximity with their parent. But it is probable that, in order to optimally utilize the unoccupied space, the nodes in one branch protrude into the neighborhood of another branch, resulting in a less structural display.
- *Bubble Tree* To heighten the sense of grouping structure, Bubble Tree [29] further constrains the proximity mapping between child and parent nodes—the child nodes are aligned in a circle around their parent, as shown in Fig. 2d. This sacrifices potential drawing area on the canvas (still more space-filling than the traditional

node-link diagram), but gains the representation of a stronger grouping structure. The user can select a branch of nodes via their parent node. However, it remains difficult to compare the sizes of the branches on the same level.

3.4 The CircleTree Exploratory Visualization

Considering the previously examined visualizations, we realize that showing the complete structure of a tree of friends may be unnecessary, even interferential to the user. As we try to facilitate the user in determining whether a friend (represented by a leaf node) can see her post, drawing too many intermediate nodes on the canvas produces unnecessary "cognitive overhead" [6], because those nodes not only occupy limited canvas space, but also increase the number of objects that the user needs to process in the limited short-term memory. Therefore, we design a new form of interactive visualization that constrains the number of levels shown (namely one or two levels) and let the user's zooming actions reveal more sub-groups or less only when she needs to. It is also similar to the Bubble Tree in the way that child nodes are positioned in a circle around their parent. We call it CircleTree.

It is important to note that in order to make visibility decisions for a post at the very beginning, the user needs to go through all the friends, regardless of the form of presentation, either simple textual list on a paper or complex visualizations. The benefit of a (good) friend grouping follows after the user's initial contact and familiarization with the generated groups. In other words, the user has made the connection between members and their corresponding group. A group is represented by a token, which can be a shape, a descriptive phrase, or the name of a member from this group, etc. This linkage information is stored in the user's long-term memory. The members can be recalled when the user just sees the group token. In such a way, the user bypasses the serial browsing of each individual member, and directly utilizes a group. The main purpose of our visualization design—CircleTree— is to provide visual tokens for a user's friend grouping. It also adds the elements of structure and engagement to an otherwise lengthy, textual reading and decision-making task. Another purpose of the visualization is to facilitate manual friend-grouping construction, as elaborated in Sect. 5. The CircleTree visualization is detailed as follows:

A node is represented by a circle, a group of nodes is represented by the circular placement of its child nodes around one extra node, which is the parent node that represents the whole circle, as shown in Fig. 3a. With the basic visual principles in mind—that humans are very sensitive to the difference of lightness in grey colors [55], we set the background color white, the friend nodes grey, the parent nodes blue. The latter two colors are also semi-transparent to avoid the occlusion effect. We pick orange and magenta as the highlight color for each friend node and parent node respectively. The large differences (from grey) in saturation and (from blue) in hue promote visual contrast [59]. At first sight, it seems sufficient to use just one visual channel to encode grouping, i.e. the circular placement of child nodes in a group. But since the user is allowed to drag the nodes to other positions on the canvas, as

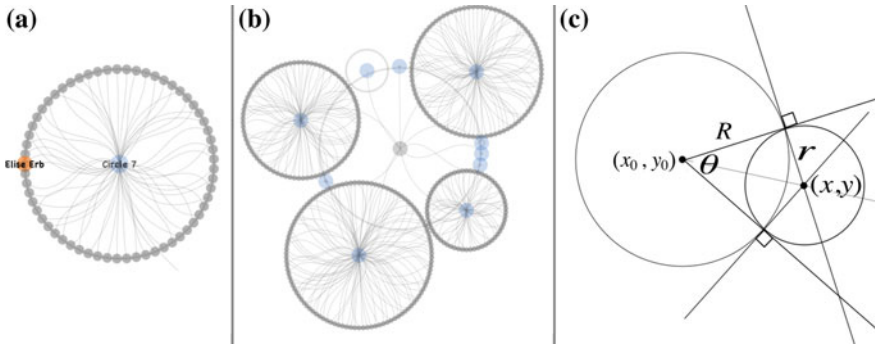


Fig. 3 **a** A single group circle, the *grey* nodes are the friend nodes, the *blue* node is the parent of the group circle. **b** The groups are positioned around a central node. **c** An illustration of drawing a group circle around a central node

described below, we add lines connecting the child nodes with the corresponding parent to emphasize that a child node belongs to its parent. The lines within a circle also signal a sense of integration. But in order to avoid overemphasizing the lines instead of the nodes, and sometimes to avoid occlusion between lines and nodes, we choose to increase the transparency of the lines.[6] Furthermore, as argued, curved lines can be used to make certain paths in a graph more apparent [61], based on [20], and curved shapes are often reflective of natural objects, giving the observer a pleasant feeling [34], we choose to use Bézier curves instead of straight lines. However, the exact role that curves play in improving the perception of grouping structure and the aesthetics of the visualization is unclear, and beyond the scope of this work.

The groups are then positioned approximately in a circle around the root node that is under focus, as shown in Fig. 3b. In the initial layout, this top node is the root of the tree. We see that the circumference of each group circle formed by its child nodes is naturally scaled with the number of children, presenting a visual order. Every pair of adjacent group circles are tangent to each other. The CircleTree layout algorithm is detailed in Algorithm 1. The radius r_i of an individual friend node from a group circle c is then approximated by $r_i \approx \pi \cdot r/|c|$, where $|c|$ is the number of friends in c, r is the radius of c. Note that very large or small m results in an exceptionally small or large r_i. Thus, minimum and maximum radii r_{min} and r_{max} are set to prevent each friend node from being too small to see or too large that it disturbs the visual ordering. When c has few friends, its assigned r becomes small, making $r_i < r_{min}$. After restoring the overly small r_i to r_{min}, we will likely have relatively large child nodes occupying the entire inner space of c and overlapping with the central parent, which does not make sense to show. Therefore such friend nodes are automatically hidden from sight, instead, the user will only see the grey circular silhouette around

[6] Note that this intended reduction of opacity does not make the lines difficult to see on a computer screen, but may lead to sub-optimal printing quality.

Algorithm 1 The algorithm for computing the layout of the group circles around a center (x_0, y_0). Note that (x_0, y_0) can be the position of the root or any center of a parent node of a group circle. We also set the maximum angle for each circle to $\pi/2$, which is an empirically derived value to keep the sizes of the generated circles contained within the canvas. For symbols θ, x_0, y_0, x, y, r and R, please refer to the illustration in Fig. 3c.

Require: the array Arr storing the sizes of the circles.
n = No.Circles, N = No.Friends, $MaxAngle = \pi/2$.
Let the array $Angles$ store the angles θ the circles.
for $i = 0$ to $n - 1$ **do**
  $Angles[i] = 2\pi \cdot (Arr[i]/N)$
  if $Angles[i] > MaxAngle$ **then**
    $Angles[i] = MaxAngle$
  end if
end for
Let the array CS store the tuples (x, y, r).
if n > 1 **then**
  $x = x_0 + |\tan(Angles[0]/2) \cdot R|$
  $y = y_0 - R$, $r = x - x_0$
  $CS[0] = (x, y, r)$
  $totalAngle = Angles[0]$
  for $i = 1$ to $n - 1$ **do**
    $r = |\tan(Angles[i]/2) \cdot R|$
    $s = \sqrt{(r^2 + R^2)}$
    $x = x_0 + \sin(totalAngle + Angles[i]/2) \cdot s$
    $y = y_0 - \cos(totalAngle + Angles[i]/2) \cdot s$
    $CS[i] = (x, y, r)$
    $totalAngle = totalAngle + Angles[i]$
  end for
else
  $CS[0] = (x_0, y_0, R)$
end if
return CS

the parent to mark the visual area of the group, keeping the visualization clean and ordered, as illustrated in Fig. 3b.

In the visualization, initially, the user only sees one layer of the tree, as an overview, but can further explore it by the zooming, panning and enabling text labels. We assume that a user can recall her impression of or her relationship with a friend if she see that friend's name. Therefore, when the mouse hovers over a node, the node is highlighted and corresponding label is shown, either a friend name or the name of a numbered intermediate node (e.g. "Circle 5" or "Circle 5.3"). Right-clicking on a parent node maps its child nodes (which we call "the focused children") in a grid layout with the names brought into sight. When a grid layout is triggered, we increase the transparency of all the other nodes on the canvas, so as to reduce the interference from irrelevant visual objects, but still keep them visible in the background to maintain a global context, as shown in Fig. 4a. Clicking (left or right) anywhere other than "the focused children" or another parent node on the canvas will

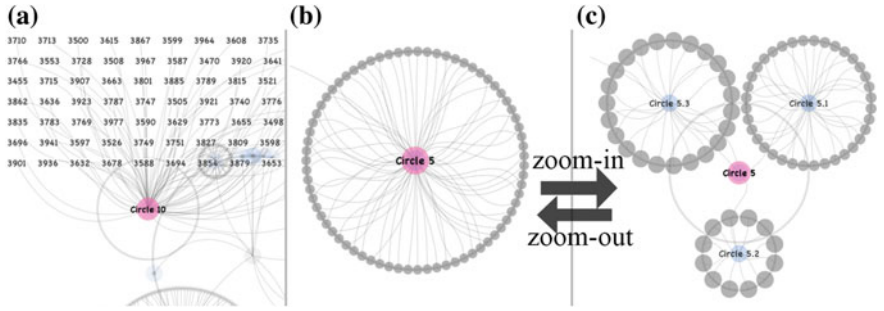


Fig. 4 **a** Right-clicking a parent node reveals the names of friends in that group. **b** A group of friends before zooming-in. **c** The same group of friends from **b** who are further grouped after zooming-in

restore the original layout. Right-clicking on another parent node will automatically restore the circular placement of the currently focused children, meanwhile shift focus onto the children of the newly clicked parent node. The user can pan (drag to displace visual objects) to adjust the point of interest. If the starting point of panning is not over a node, the whole tree will be panned. If it is over a node, that node will be panned, along with its child nodes if it is a parent.

We take the current mouse position on the canvas as the "anchor point" for zooming actions. An anchor point $P_{anchor} = (x_a, y_a)$ is the position that is invariant during zooming. A zooming action triggers the following transformation: $rt \cdot \beta \cdot (P' - P_{anchor}) = (P' - P)$, in which $P = (x, y)$ is the position before zooming, $P' = (x', y')$ is the position after zooming, $rt \in \mathbb{R}$ is the value of mouse-wheel rotation provided by the operating system, $\beta \in \mathbb{R}$ is a constant adjusting the zooming speed. Note that the zooming speed on X- and Y-axes are the same. It then follows that the scaling factor is $sf = (x' - x_a)/(x - x_a) = (1 - rt \cdot \beta)^{-1}$. During zooming, the radius r_i of each node is multiplied by sf but further constrained by $r_i \in [r_{min}, r_{max}]$. When the child nodes no longer overlap with the corresponding parents, the hidden child nodes and their names are brought into display with zooming-in. When the user zooms into one circle of friends, we perform a "focus-check" to determine whether to further divide the circle. The "focus-check" assumes a rectangular area, half the width and height of the canvas, with the current mouse position as the center point. Upon the user's zooming-in, the only remaining group circle whose parent node is within this area is found and divided. The newly generated sub-circles are presented if the subgraph corresponding to the circle is divisible according the algorithm [46]. We choose the size of the "focus-check" area such that the user does not need to zoom too deeply or too shallowly to explore sub-circles. Zooming out of the visualization makes the sub-circles from the previously divided circle squeezed and overlapped, which will trigger them to merge back to the singular circle again. This is depicted in Fig. 4b, c.

3.5 Participants, Tasks and Results

In this section, we document the tasks of the participants and the findings from the study.

3.5.1 Participants and Tasks

There were 16 participants (three females), 25–45 years old, from eight countries. Among them were Ph.D students, company employees and graduate students. The participants were equally divided into two groups, which we named directly with the corresponding algorithm abbreviations mentioned in Sect. 3.2—HMOD and FSL. Both groups used the same visualization interface detailed in Sect. 3.4, but with different community detection methods, as their names suggested, namely the hierarchical modularity-based algorithm and Facebook smart lists. Because the latter was not a complete grouping, the friends of a participant that were not in any smart list were put together as one other group.

Our assumption in the user study is that users utilize categories of friends (denoted as C_u) to make a binary visibility decision. We denote the communities that HMOD and FSL produce as C_{HMOD} and C_{FSL} respectively. C_{HMOD} is the result of the interactions between a user and HMOD, with the CircleTree visualization interface. C_{FSL} is the set of non-hierarchical circles of friends constructed from the user's Facebook smart lists, with one extra circle containing the friends who are not in any of the smart lists. Our hypothesis is that, for users' visibility decision-making, C_{HMOD} coincide with C_u, more than C_{FSL}.

We asked the participants to perform the following two tasks: elicitation of regrets in posts and visibility decision-making. In the first task, each participant was asked to identify her regretted posts. In the second task, the participants in the two groups HMOD and FSL were asked to make visibility decisions for each of their posts. Each group has 8 participants and 24 posts. As illustrated in Fig. 5, when a participant

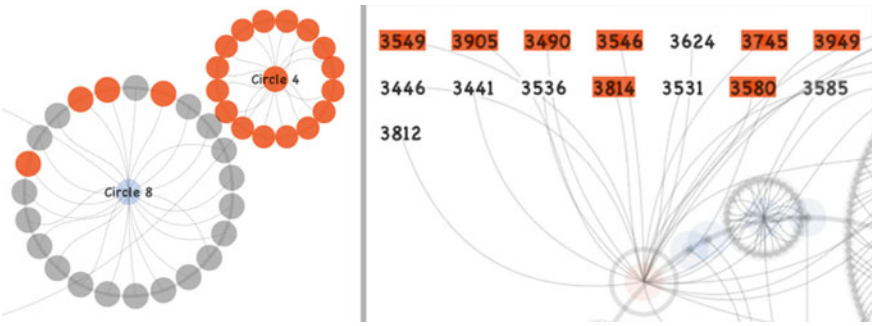


Fig. 5 Participants can determine a post's visibility to each friend individually by clicking friend nodes or collectively by clicking parent nodes in the centers of group circles

thinks a friend can see the post, she clicks on the corresponding friend node, the color of which changes to indicate that the post is now visible to the clicked friend. Clicking on the parent node of a group circle toggles every child node's color, or further descendants if some child nodes are already divided by a zooming-in action. The participants could work at their own pace until they were satisfied with their decisions.

Though recent studies have investigated regrets in OSN from different aspects [41, 60], we chose to let the participants explicate their own regrets, as it is easier for a person to make visibility decisions based on her own experience. We collected the posts in face-to-face interviews with the participants. We emphasized the difference between complete and partial regrets. A complete regret meant that the post was supposed to be seen by no one. A partial regret meant that the participant did not mind her post being seen or intended her posts to be seen by some of the friends, but failed to block the other undesired friends. Since a complete regret entails concealing the corresponding post completely, which would render a visibility decision trivial, we guided the participants to only think of partial regrets. Each participant was encouraged to think of at least three posts. A post needs to be specific enough to let the participant define its visibility towards each friend. In total, 48 posts were collected; each participant contributed three personal posts on average. We found that photo-related posts were mentioned frequently, thus making a distinction between photos and topics. Topic-related posts include status updates, web-link sharing and comments.

We recorded the participants' regretted posts and manually classified them into five categories, as summarized in Table 1. The *first* category covers the posted photos that cause embarrassment or awkwardness, typical examples are "drunk party" photos. There are also the photos showing the participant together with some particular person(s), e.g. ex-boy/girl-friend, that the participant feels the need to hide the photos from some friends. The *second* category covers the photos that are less sensitive in terms of embarrassment or awkwardness, but still in need of visibility control. For example, some photos may be so intimate that the participant only wants to show them to her family and best friends. Some photos were taken at a event with a specific group of people, only to whom, as participants argues, the photos should be made visible. More than a third of the posts are photo-related. The *third* category covers the topic-related posts that involve explicit self-expression, including strong opinions and emotional expressions, such as venting negative emotions. Of the seven

Table 1 Participants' regretted posts

	Categories of regretted posts	Frequency
(1)	Sensitive photos causing embarrassment or awkwardness	8
(2)	Other photos for a specific group of friends	9
(3)	Sensitive topics involving emotional expressions	7
(4)	Sensitive topics involving nasty jokes	12
(5)	Other topics for various specific situations	12

posts in this category, six are about venting or expressing negative opinions, which the participants felt should be avoided in future, for those posts may harm one's image if disclosed carelessly. The *fourth* category covers the sensitive topics that are less self-involved, but more about the intrinsic sensitive nature of the content of the posts, including politics, religion, sex, race and/or nasty jokes. It is interesting to see that nine out of the twelve posts in this category are about inappropriate jokes. For example, several participants reported that they posted something they believed sarcastically humorous, but in hindsight, they thought it was not wise to expose those posts publicly, as some friends may not understand the humour, or even be offended by it. The *fifth* category covers the relatively less sensitive topic-related posts, which nonetheless need visibility control. For instance, it may not make sense to show the posts to the friends who do not speak the language in which the posts are written.

3.5.2 User Study Results

We use binary entropy to evaluate the effectiveness of the two approaches for users making visibility decisions. $Entropy(post) \in [0, 1]$ (Eq. 1) calculates the information content (in bits) needed to determine whether a member in a circle can see a post. C is a set of circles of friends, and $c \in C$ generated by HMOD or FSL. $V_{c,post}$ is the number of the friends to whom *post* is visible in the circle c. N is the total number of friends (including duplicates if circles overlap) in all the circles. $Entropy(post) = 1$ means that on average, in one circle, half the circle can see the post while the other half cannot. This indicates that the given set of circles is unhelpful for the user to make visibility decisions on a group-level, by taking the circles holistically into account. $Entropy(post) = 0$ means that for each circle, the friends in the same circle have the same visibility access to the given post. That is, every circle can be fully utilized by the user to make visibility decisions. The CircleTree visualization in the group HMOD is analogous to a binary-classification tree. Users try to use this tree to make visibility decisions. A"pure" circle in terms of visibility decisions is helpful, since such a circle can be considered as a whole. The initial circles are divided until they are indivisible according to the graph modularity or they are pure. Then the sub-circles are used to calculate entropy scores.

$$\begin{aligned} Entropy(post) &= \sum_{c \in C} \frac{|c|}{N} Entropy(c, post) \text{ with} \\ Entropy(c, post) &= -\frac{V_{c,post}}{|c|} \cdot log_2 \frac{V_{c,post}}{|c|} - \frac{|c| - V_{c,post}}{|c|} \cdot log_2 \frac{|c| - V_{c,post}}{|c|} \end{aligned} \tag{1}$$

Another aspect of a set of visibility decisions for a user's post is its imbalance. That is, the number of friends who can see the post is significantly different than those who cannot see the post. Let V_{post} be the total number of friends who can see the post and $\alpha = min(V_{post}, N - V_{post})$. When α is rather small, e.g. one or two, $Entropy(p)$ can be low almost regardless of which grouping method is used. In such case, while

Table 2 Entropy scores for group FSL and group HMOD, with $\alpha > 1$ and $\alpha > 5$

	FSL	HMOD
$\alpha > 1$ (24 posts)	**0.46**	**0.20**
$\alpha > 5$ (19 posts)	0.56	0.22

a grouping may still be useful for the participants to browse friends, but it is likely to be less effective for making visibility decisions than the participants just typing individual friend names to search for them in real-time, as discussed in Sect. 3.1. We know that the average number of friends of each participant is 194. All the 48 posts (24 posts for each group) have $\alpha > 1$ and $\overline{\alpha} \approx 34$. Within these posts, there are 38 posts (19 posts for each group) with $\alpha > 5$ and $\overline{\alpha} \approx 42$. Table 2 shows the average Entropy scores in group HMOD and FSL for $\alpha > 1$ and $\alpha > 5$. Group HMOD achieves lower entropy than group FSL in both cases. This suggests that the circles generated by the hierarchical modularity-based method are taken more holistically into consideration than Facebook smart lists by the participants to make visibility decisions. In other words, it is more often that a circle in the HMOD group, than that in the FSL group, is marked unanimously as the people who "can see" or "cannot see" a post. We can also see that raising α level indeed increases the average entropy scores in both groups, but the increase is more apparent in group FSL ($\approx 22\,\%$) than in group HMOD ($\approx 10\,\%$).

We test the statistical significances of the differences between the entropies from HMOD and FSL. It is however, less straightforward to compare the two, because the entropy scores yielded in group FSL are from a different set of participants, with a different set of EOSN and posts. Nevertheless, it is possible to perform an approximate comparison by a pessimistic pair-wise matching. We first calculate the pair-wise squared entropy differences between FSL and HMOD/MOD, deriving a cost matrix, with which, we match the entropies in the two groups via the linear assignment [43] to minimize the sum of the pair-wise differences (so as to minimize the difference between the two models). Based on the resulting pair-wise matches, we perform the t-tests. It then follows that, in comparing HMOD and FSL, the t-statistic is 9.146 for $\alpha > 1$ and 12.810 for $\alpha > 5$. The t-statistics reject the corresponding null hypotheses with two-tail Confidence Interval $(CI) = 99.9\,\%$ and one-tail $CI = 99.95\,\%$. It is then evident that HMOD is significantly better than FSL.

From this user study, we gain more insight into users' privacy decision-making process in online posting. *First*, it is evident that categorical thinking is used when users make binary visibility decisions. *Second*, graph-modularity-based friend communities assist users more efficiently for such decisions than the profile-attribute-based Facebook smart lists. This implies that the former produces the communities that fit the categories of friends that a user has in mind, more than the latter. We examine another state-of-the-art community detection algorithm for EOSN in comparison with the modularity-based algorithm in Sect. 4 and discuss the implications of the results. Third, the essence of categorical thinking is to reduce the cognitive load. If there are too many information objects (in our case, online friends), hierarchical

categorization supports the users' visibility-decision-making. When the resolution limit of MOD is reached, the sub-circles can be of more help for the users. The results also give us guidance in designing information visualization systems—categorization and abstraction are important for users to process large amount of information.

4 Community Detection and Social Groups

In this section, we first introduce the two community detection methods of interest (Sect. 4.1), then describe the datasets (Sect. 4.2) on which the two algorithms run. We compare and discuss the performances of the two algorithms on these datasets (Sect. 4.3), and propose an extension of one of the algorithms to accommodate the overlapping nature of online friend groups (Sect. 4.4). We summarize and compare the performances of all three algorithms by the end of this section.

4.1 Two Models for Community Discovery

From our preliminary user study, we know that, in order to make sense of the friends in one's online social life, it is important to categorize them, either for the ease of processing and memorizing friends' information, or as an efficient means for making decisions. Given the large number of friends that one usually has in EOSN, automated community detection can be very helpful not only as the basis for visibility decisions, as investigated in the previous section, but also for other tasks in online contact management, such as simply keeping an overview, sorting incoming messages, etc. In this section, we examine community detection algorithms and their relationship with real-life social groups. We compare two models for community detection in EOSN: the graph-modularity-based model (MOD) using eigenvalue decomposition [45] and the Generative Model for Friendships (GMF) [39].

Modularity is the number of edges falling within groups minus the expected number in an equivalent network with edges placed at random [46]. Larger modularity value suggests more obvious community structure in the graph. There exists abundant and different techniques that optimize the modularity of a graph. We chose to implement Newman's spectral optimization algorithm that iteratively bisects a given graph using the eigenvectors of the modularity matrix. This approach is generally more accurate than the techniques such as greedy methods and external optimization, and less computationally expensive than global optimization approaches such as simulated annealing [21]. We also implement vertex-moving to improve the final modularity score, as proposed in [46]. Intuitively, in each bisection of the input (sub-)graph, "vertex-moving" moves one vertex at a time, from one (sub-)community to the other, if the modularity is increased, it makes this move permanent. The average, combined complexity of this algorithm is $O(N^2 log\ N)$.

GMF is a recently proposed community detection model that leverages both the friend-profile features and the friend-graph structure in an EOSN [39]. The resulting communities have the following properties: (1) the friends in the same communities have common features, such as education, work; (2) different communities may emphasize different features; (3) the communities may overlap. GMF has been evaluated against the ground-truth communities from three EOSN datasets (as described in Sect. 4.2), and compared with eight baseline models—Mixed Membership Stochastic Block Models, Block-LDA, K-means clustering, Hierarchical Clustering, Link Clustering, Clique Percolation, Low-Rank Embedding and Multi-Assignment Clustering (as elaborated in [39]). It was demonstrated that GMF generated more accurate communities than the baselines.

4.2 Three EOSN Datasets

The three datasets were collected from Facebook, Twitter and Google+, which are available online.[7] We downloaded these datasets, removed empty files, and discarded the ego-networks whose ground-truth circle(s) contains just one friend. Finally we obtained 10, 909 and 129 ego-networks from Facebook, Twitter and Google+ respectively, which we use for our experiments. Note the data is a subset of the data used in [39]. Each ego-network includes the user's and the friends' profiles, the friend graph and the set of manually constructed circles by the user. For the Twitter and Google+ friend graphs, we ignore their directivity as MOD runs on undirected graphs. We denote the complete set of friends as V, the friend nodes retrieved from the user's ground-truth circles in an EOSN as $V_{circles}$, the friend nodes retrieved from the user's friend graph as V_{edges}, a ground truth circle as c, the set of ground-truth circles as C, an algorithm-generated circle as c' and a set of algorithm-generated circles as C'. The three datasets are summarized in Table 3. We see that $\overline{|V_{circles}|} < \overline{|V_{edges}|}$ for the three datasets, since $V_{edges} \subseteq V$, it indicates that $V_{circles} \subset V$. Moreover, we observe that overlapping ground-truth circles are common, but also limited such that a friend is usually assigned to less than two circles.

Table 3 Three ego-network datasets summarized, from left to right

EOSN	$\overline{\lvert V_{circles}\rvert}$	$\overline{\lvert V_{edges}\rvert}$	$\overline{\lvert C\rvert}$	$\overline{\lvert c\rvert}$	$\overline{No.Comms.P}$
Facebook (10)	298	423	19.3	26	1.6
Twitter (909)	36	134	4.4	12	1.4
Google+ (129)	304	1948	3.6	135	1.6

$\overline{|V_{circles}|}$ is the average number of friends from a user's ground-truth circles, $\overline{|V_{edges}|}$ is the average number of friends from a user's friend graph, $\overline{|C|}$ is the average number of a user's ground-truth circles, $\overline{|c|}$ is the average ground-truth circle-size, $\overline{No.Comms.P}$ is the average number of ground-truth circles to which a friend belongs

[7] https://snap.stanford.edu/data/index.html#socnets [Accessed on Dec 9, 2013].

4.3 Performances of GMF and MOD

We follow the same method and metrics in [39] to evaluate how well a set of generated circles C' match the user's manual circles C. Balanced Error Rate (BER) [12] and F1 scores are used to measure the matches of circles, as defined in Eqs. 2 and 3. We use $RBER(c, c')$ to refer to $1 - BER(c, c')$. In order to determine which $c' \in C'$ corresponds to which $c \in C$, we perform a linear assignment using the Hungarian Algorithm [43] to maximize the sum of the pair-wise *RBER* or $F1$.

$$BER(c, c') = \frac{1}{2}\left(\frac{|c \backslash c'|}{|c|} + \frac{|c' \backslash c|}{|V_{circles}| - |c|}\right) \tag{2}$$

$$F1(c, c') = 2\frac{|c \cap c'|}{|c| + |c'|} \tag{3}$$

We ran GMF[8] and MOD on the ego-networks that only included the friend nodes from ground-truth circles, so that we could compare C and C'. The reason that we ran GMF again instead of directly using its original result was because of the incomplete ego-network data that we could download and some trivial data (e.g. an ego-network containing only one friend) that we discarded afterwards. As such, both GMF and MOD were run on the subsets of the ego-networks that were described in [39], namely 10 Facebook, 909 Twitter and 129 Google+ ego-networks instead of 10, 1,000 and 133 ego-networks. Due to the complexity of the algorithm (with the worst case complexity $O(N^3)$, N being the number of friend nodes in an ego-network), we ran GMF for each ego-network with selective K values (the number of communities), $K = 3, 5, 7$ and 9 respectively. Then we select the K value that corresponds to the highest average $\overline{RBER}$ or $\overline{F1}$, and match C and C' for each ego-network via linear assignment. As for MOD, K is automatically derived in the process of modularity maximization. The results are summarized in Tables 4 and 5. Note that while certain K of GMF achieves the highest $\overline{RBER}$, it does not necessarily mean this K corresponds to the highest $\overline{F1}$. Thus we have two different sets of combinations of Ks with respect to the $\overline{RBER}$ and $\overline{F1}$ measures. The columns $|\overline{C'}|$ and $\overline{No.Comms.P}$ in Table 5 are based on the average values of these two sets of Ks.

Table 4 The comparison between the results of GMF running on the subsets with four K choices (*white columns*) and the original sets (*gray columns*) of the ego-networks: Facebook (Fb), Twitter (Tw) and Google+ (Gp)

GMF	Fb(10)	Fb(10)	Tw(909)	Tw(1000)	Gp(129)	Gp(133)
$\overline{RBER}$	0.83	**0.84**	**0.77**	0.70	0.65	**0.72**
$\overline{F1}$	0.53	**0.59**	0.32	**0.34**	0.24	**0.38**

[8] The code can be downloaded from the author's web page: http://i.stanford.edu/~julian/. We used the default parameters in the code with different K values.

Table 5 The results of running GMF and MOD on the three subsets of ego-networks

EOSN	$\overline{RBER}$		$\overline{F1}$		$\overline{\lvert C' \rvert}$		$\overline{\lvert c' \rvert}$		$\overline{No.Comms.P}$	
Facebook	0.83	**0.86**	0.53	**0.67**	3.3	7.0	90	41	1.5	1
Twitter	0.77	**0.81**	0.32	**0.68**	5.2	3.0	7	12	2.7	1
Google+	0.65	**0.75**	0.24	**0.62**	6.9	3.1	44	98	3.8	1

The *gray sub-columns* are the results for GMF, the *white ones* are for MOD. $\overline{|C'|}$ is the average number of generated circles, $\overline{|c'|}$ is the average size of each generated circle, $\overline{No.Comms.P}$ is the average number of circles to which each friend belongs

We denote the GMF algorithm that was run on the original ego-network datasets, with a full range of K values checked, as GMF0. This is to differentiate it from the GMF model that we ran on the subsets, with the four K values checked. From Table 4, we notice that the *RBER* and $F1$ scores of GMF on the Facebook and Google+ datasets are smaller than those of GMF0, and the *RBER* and $F1$ scores of GMF on the Twitter dataset are comparable to or higher than those of GMF0. The relatively large performance difference on Google+ is due to the limited choices of K in GMF. From Table 5, we see that MOD fully outperforms GMF on *RBER* and $F1$ measures.

4.4 Multi-membership Modularity-Based Method

From Table 5, we can also see that MOD generates the $\overline{|C'|}$ that is closer to the ground-truth as shown in Table 3. We also know that though overlapping circles are common in the ground-truth, one friend is rarely put into more than two circles, whereas GMF on Twitter and Google+ generates the circles that have $\overline{No.Comms.P}$ equal to or larger than three, which led to its relatively low performance on these datasets. However, a significant limitation of MOD is that it produces non-overlapping communities, while it is obvious that OSN users construct overlapping circles by themselves. Thereby, we propose an extension of MOD that allows multiple circle memberships, which we call Multi-membership Modularity-based community detection, shortly as MMOD. We define a metric we call the External Belongingness (EB as in Eq. 4), in which $neighbors(v, c')$ is the number of neighbors (one hop away on the friend graph) of a given friend v in an external circle c', $degree(v)$ is the degree of v. c' is external to v if $v \notin c'$. We first run MOD to derive a set of non-overlapping circles. Then for each friend, we obtain a list of external circles (the circles to which the friend does not belong) with the corresponding EB scores. We subsequently check the highest EB score for each friend, if it exceeds the previously defined θ_{EB}, the friend is assigned to the corresponding external circle. In this way, we obtain a set of overlapping circles with some friends belonging to two circles. However, it remains the question of how to select θ_{EB}. We run MMOD with different $\theta_{EB} \in [0, 0.5]$ with the step size 0.05. Then we match the respective overlapped C' with C, the performances are

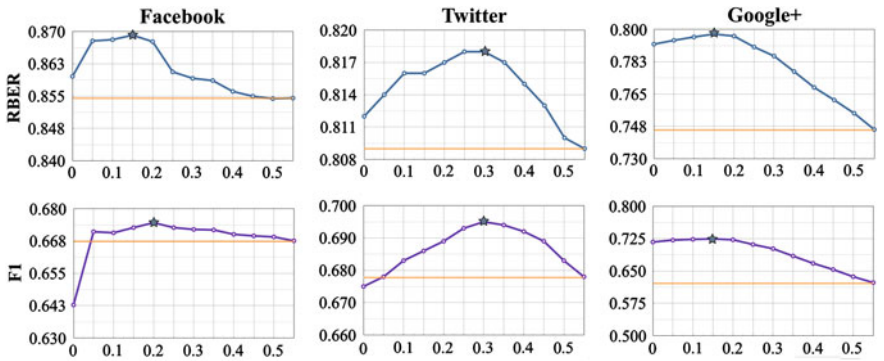


Fig. 6 The *RBER* and $F1$ performances of MMOD with different θ_{EB} values. The baselines are drawn to indicate the corresponding MOD performances and the stars are to mark the optimal θ_{EB} points

Table 6 The results of running MMOD on the three subsets of ego-networks

EOSN	$\overline{RBER}$	$\overline{F1}$	$\overline{\lvert C' \rvert}$	$\overline{\lvert c' \rvert}$	$\overline{No.Comms.P}$
Facebook	0.87	0.67	7.0	52	1.3
Twitter	0.82	0.70	3.0	18	1.5
Google+	0.80	0.73	3.1	166	1.7

$\overline{|C'|}$ is the average number of generated circles, $|c'|$ is the average size of each generated circle, $\overline{No.Comms.P}$ is the average number of circles to which each friend belongs

plotted in Fig. 6. In each plot of Fig. 6, the last point is the average *RBER* or F1 score from MOD, through which a straight horizontal line is drawn to indicate baseline performance. The point with the highest performance is marked with a star.

$$EB(v, c') = \frac{neighbors(v, c')}{degree(v)}, v \in V, v \notin c' \tag{4}$$

From Fig. 6, we can see that the performances of MMOD are generally better than those of MOD. The curves also follow the similar trend that increases till some particular θ_{EB} and drops. Around $\theta_{EB} = 0.5$, rarely any friend nodes can be found in external circles, thus the performances regress to be close to MOD's. We also find that the optimal threshold θ_{opt} values for Facebook and Google+ data are similar, which stay around 0.15 for both *RBER* and $F1$, whereas for Twitter data, this value is 0.35. The *RBER* and $F1$ scores of MMOD at these θ_{opt}, along with other results (the same columns as Table 5) are summarized in Table 6, from which we see that MMOD fully outperforms MOD, and that the $\overline{No.Comms.P}$ values are very close to those of the ground-truth datasets. The better results on MMOD also have the implication that people indeed tend to put the friends who are the connectors or hubs in the ego-network into different circles at the same time. We summarize the performances of GMF, MOD and MMOD in Fig. 7.

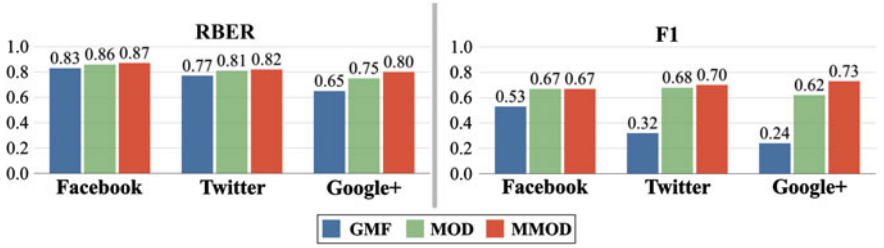


Fig. 7 The overview of the performances of GMF, MOD and MMOD

We also observe that θ_{opt} empirically correlates with the average size $\overline{|V_{circles}|}$ of an ego-network, which is around 300 on Facebook and Google+, and 30 on Twitter. For instance, we can describe this relation with Eq. 5. If we consider the MMOD-generated circles match the user's manual circles better (indeed, the *RBER* rates are close to or well above 0.8), the relation in Eq. 5 suggests that, on the one hand, users tend to manually create less overlapped circles when they have fewer friends. On the other hand, θ_{opt} decreases exponentially slower than the number of one's friends increases, which means that on a relatively large scale (e.g. $|V_{circles}| \in [100, 1000]$), given that EOSN are often sparse [42, 57], users' θ_{opt} for allowing a friend to be in multiple circles remains similar ($\theta_{opt} \in (0.12, 0.20)$ approximately). However, in order to accurately capture the relationship between the number of friends and the optimal threshold, we need a further investigation. It may involve other potentially correlated parameters, more sophisticated models and more data, which is beyond the scope of this work. Equation 5 is manually derived based on the observations from Table 3 and Fig. 6. It serves as an intuitive guidance for determining θ_{opt}.

$$|V_{circles}| = 3 \times 10^{(\frac{0.3}{\theta_{opt}})} \Longleftrightarrow \theta_{opt} = \frac{0.3}{lg|V_{circles}| - lg3}, \ \theta_{opt} > 0 \tag{5}$$

We perform ANOVA (ANalysis Of VAriance) to compare GMF, MOD and MMOD on the three datasets. The **p** values are summarized in Table 7. We can see that the **p** values on the Facebook dataset are rather high, and the **p** values on the other two datasets are low ($\mathbf{p} < 0.001$). This means that the variance between the three models is not significant on the Facebook dataset, but very significant on the Twitter and Google+ datasets (in fact, the F-statistics on these two datasets approach the ends of the corresponding F-distribution curves.) In Fig. 7, the observed differences were statistically significant for both $\overline{RBER}$ and $\overline{F1}$ on the Twitter and Google+ datasets (all $\mathbf{p} < 0.001$ for one-way ANOVAs), but not for the Facebook dataset ($\mathbf{p} = 0.43$ for $\overline{RBER}$ and $\mathbf{p} = 0.13$ for $\overline{F1}$). The latter may be a result of the small sample.

Table 7 The **p** values of the ANOVA for GMF, MOD and MMOD, of both $\overline{RBER}$ and $\overline{F1}$measures, on the datasets of Facebook, Twitter and Google+ respectively

	Facebook	Twitter	Google+
$\overline{RBER}$	0.43	<0.001	<0.001
$\overline{F1}$	0.13	<0.001	<0.001

4.5 Discrepancy Between Predicted and Manual Circles

Though a community discovery algorithm can predict reasonably good circles, it is unlikely that it can make a perfect prediction. This attributes to the fact that manual circle-creation process is inherently subjective, and varies on the same person for different purposes. The ground-truth circles of the ten Facebook users that we used in our experiments were obtained by a Facebook app,[9] in which the user entered comma-separated category labels for each friend. Existing labels could be reused by a selection from a drop-down box. Each label represented a circle to which a friend belonged. The text cue for entering the label(s) for each friend **Fr** was "I know **Fr** because ..." followed by the label-entering text-field. In another exercise [15] of friend-grouping, the groups (i.e. circles) were constructed by "card sorting". The name of each friend of a participant's was printed on a paper card. Several cards were randomly selected and spread on a table, the participant was then asked to assign the rest of the cards to the selected ones to form groups. We can see that the Facebook app friend-grouping exercise encourages more overlapping circles to be created than the card-sorting exercise.

Different user interfaces may directly reflect intrinsic and systematic differences on a functional level, rather than on a perceptual level. Facebook provides a social platform mainly for mutual friends—two people become friends when one "accepts" the other's "friend request". The friends of friends are recommended if the user wants to add more friends. Twitter and Google+ implement a "follower-followee" mechanism, which means a friendship is not necessarily reciprocal. On Twitter, the user clicks the "follow" button to follow a "friend", every newly followed friend is not necessarily put into a friend list (i.e. a circle), whereas on Google+, the "follow" button becomes the "add" button, and every newly followed friend has to be added into one of the existing circles or a new one. This is an important reason that the number of friends in Google+ circles is much more than that in Twitter lists, as shown in Table 3. We see that people create circles differently under different circumstances, consciously or unconsciously. It is therefore important to create interfaces that help users gain insights about their EOSN friendships from different aspects, and let them form their own friend circles with more informed decisions.

Moreover, social and cognitive theories shed light on human social grouping behavior and inform computer scientists to design community detection algorithms

[9] https://www.facebook.com/apps/application.php?id=201704403232744 [Accessed on Dec 12, 2013].

and interactive visualizations. The social brain hypothesis (SBH) offers a framework for integrating evolutionary and social psychological perspectives on human social complexity. SBH predicts a natural community size of around 150 for modern humans (Dunbar's number [17]), and now there is considerable evidence confirming that this is the typical size of both personal social networks and key types of human community [18]. Note that 150 is the typical size of a person's active network, in which she knows how these the friends fit into her social world and they know how she fits into theirs [18]. From the literature in cognitive science, we also know that there is the cognitive capacity limit in human Short-Term-Memory (STM), which is inline with the theory of categorical thinking (Sect. 3.1). This capacity limit is averaging on seven [40], which means that people can remember seven chunks of information in STM tasks. In our case, we can consider a chunk to be a group of friends. This limit is subject to debate, later evidences showed that it was a high estimate, lower numbers were proposed, e.g. four [14]. The theories on social group size and human's cognitive capacity limit provide more incentives for interactive visualizations, which should enable users to flexibly interact with friend visual objects on different granularity-levels—from (sub-)groups of friends to individual friends.

5 Improving the Tool Design

From the previous sections, we understand that grouping friends is important for OSN users to manage online contacts and make privacy decisions. A carefully designed community detection algorithm can produce decent friend circles that match users' manual circles, but this matching is hardly perfect due to the subjective nature of friend grouping. To close the gap between computer-based grouping and human grouping, tools need to be designed and built. The goal of such tool that leverages interactive visualization and accurate community detection is not only to show its users their structured friendships, but more importantly, to make the structures more usable for the users.

We have introduced a tool in our user study to assist users' visibility decision-making (Sect. 3). It visualizes the generated circles of the user's friends, and allows hierarchical exploration. However, this tool addressed only part of the information about the user's friends, with limited navigation functions. As various taxonomies for visual analytics or information visualization unanimously emphasized [25, 32, 53, 63], presenting multiple aspects and providing multiple perspectives are essential for visualizing large and complex data. We developed a new online application named FreeBu.[10] We motivate and describe three more views that supplement the CircleTree view (Sect. 3.4). All the four views serve a two-fold purpose: (1) to provide users with different insights about their own ego-networks, (2) users can manually construct their Facebook friend lists with the tool.

[10] http://people.cs.kuleuven.be/~bo.gao/freebu/ [Accessed on Dec 12, 2013].

From Sect. 3 we have known that FSL are less efficient in visibility decision-making than the communities generated by MOD, suggesting that graph-based communities coincide more with the friend groups that a user has in mind. An investigation (Sect. 4) in the three community detection algorithms GMF, MOD and MMOD has shown that the ground-truth circles are still in favor of the graph and modularity-based methods. And introducing overlaps further increases the *RBER* and $F1$ accuracies. However, it is incorrect to assume the human friend-grouping process is systematically similar to the algorithmic process just because both produce similar groups. The CircleTree visualization shows circles as friend groups, in which each member is labeled by the name. Other types of data, such as profile, posts, chat history, friend graph, etc. are also potentially useful for the user's understanding of her own ego-network. They can inspire the user to reflect on her online contacts and facilitate friend-group creation.

As the recent study [15] on OSN-friend-grouping shows, people do consider attributes, such as school, music band or youth community, when they group friends, we refer to this type of grouping strategy as the *Attribute* strategy. Also, people indeed tend to put the friends who are mutually friends into the same the group, we refer to this strategy as the *Graph* strategy. Another graph-related, but slightly different grouping strategy is based on some particular friends—"I know those friends via this friend", we refer to it as the *Connection* strategy. The fourth strategy is based on trust or closeness, to which we refer as the *Closeness* strategy. Informed by these grouping strategies, we have four visualizations in FreeBu to accommodate users' comprehension of their online friends and help users create friend groups semi-automatically. The four visualizations/views are described as follows:

- *Circle View* The circle view (i.e. the CircleTree Visualization) is for the *Graph* strategy. Mutually connected friends tend to be put in the same circle, the user can drag and drop a circle or an individual node to compose her own Facebook friend list (Fig. 8a). The group circles with different sizes also provide the user with a sense of ordering, helping her quickly find outliers or surprising circles. The visualization and interaction strategies are detailed in Sect. 3.4.
- *Map View* The map view is for the *Graph* and the *Connection* strategies. The user's friend graph is directly visualized in a force-directed layout with the Fruchterman-Reingold algorithm [23], which is a typical graph-layout algorithm. It pulls connected nodes together and pushes disconnected nodes apart. Users can easily observe visual clusters and hub-nodes. The user can zoom and pan to explore the graph, zooming-in brings out the node labels. Mouse-hover on a friend node also brings out the friend's name label, meanwhile highlights the connections of this friend on the graph. Right-clicking a friend node will automatically select this node as well as its neighbors. User can then drag and drop the selected nodes to compose her own friend list (Fig. 8c). Furthermore, the nodes' radii are set proportionally to the corresponding Betweenness [44] scores, so that the important nodes that connect different parts of the user's ego-network are enlarged and emphasized. It has been shown that the bridging structure in a user's EOSN is important for predicting strong social ties, such as romantic partners [4]. To make the

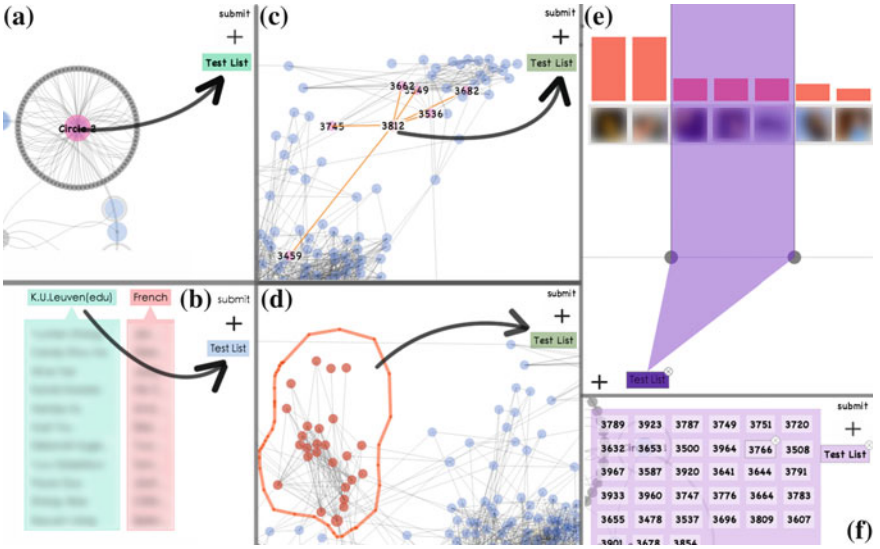


Fig. 8 This figure shows the four views in FreeBu and the drag-drop actions to compose user-defined lists. Each arrow indicates a group-level drag-drop action. **a** circle view, **b** column view, **c** map view: immediate-neighbor selection, **d** map view: custom selection, **e** rank view, **f** list-editing

group-creation more flexible, the point-in-polygon function is implemented. The user can turn on this function by pressing the "pen" button on the bottom-right corner of the canvas and draw a polygon to enclose and select the nodes of interest, and drag-drop the selected nodes to compose lists (Fig. 8d).

- *Column View* The column view is for the *Attribute* strategy. The column view generates the groups of friends based on common profile-attributes between friends, which is a generalization of Facebook smart lists. Each column represents a group. The "head" of the column is labeled with the corresponding attribute-value name. The "body" is a stack of friend name tags belonging to that column. If a column contains more than N_{col} (e.g. $N_{col} = 12$), only N_{col} friend tags are initially shown in the body of the column, with the "..." symbol to indicate there is more tags. Mouse-hover on the head of a column expands the column and show all the member names. The heights of the columns are proportional to corresponding the numbers of friends. Users can scroll left or right with mouse wheel to explore the columns. They can click the "overview" button for a summary of all the column labels. The user can drag and drop a column or an individual tag to compose lists (Fig. 8b). Moreover, the user can drag and drop columns into the "intersection" area at the bottom of the canvas. This area keeps the members that satisfy the attribute values from the columns. The user can then use intersected area (also via drag-drop) to compose her friend lists.
- *Rank View* The rank view is for the *Closeness* strategy. Studies [18, 56] have shown that interaction frequency linearly corresponds to the strength of interpersonal ties.

We visualize the users' friends by aligning their profile photos horizontally near the middle of the canvas. The photos are ranked according to the communication frequencies of the user with her friends in Facebook chat. On top of each photo, a bar is shown if there is a communication history of the user with that friend. The more frequently the user chatted with a friend, the higher the bar is. The user can scroll left or right with mouse wheel to see the bars and photos. Mouse-hover can enlarge a photo can brings out the name beneath it. The user can select one or more friends by moving the two "knobs" with vertical lines. Clicking on a user-defined list "absorbs" the friends that are "clipped" by the two knobs into that list (Fig. 8e).

The four views share a similar way for creating customized friend lists. The user starts by clicking the "plus" button to add a new, empty list, aligned on the right (in the first three views) or the bottom (in the rank view) of the canvas. Each list is shown as a rectangle. The user can right-click a list to edit its name. Drag-drop actions put selected friends into a list, as illustrated in Fig. 8a–d, whereas in the rank view, "clipped" friends are put in a list by user clicking on the list, as shown in Fig. 8e. Mouse-hover on a list brings out the friend-name tags of the list in a grid layout. Mouse-hover on a list or a tag also brings out the "remove" button, as shown in Fig. 8f. In this way, the user can remove a list or a member if needed. The user can submit the lists to her Facebook account by clicking the "submit" button.

An elaborate multi-method user study on the usefulness and perceived values of FreeBu is beyond the scope of this chapter. We refer to [16] that has detailed such study. Through a factor analysis, it showed that FreeBu received high scores (between 4 and 6 on a 7-point Likert Scale) on several factors of perceived values. These factors include Audience Control and Audience Reflection. The first factor refers to sharing information with differentiated friends. For example, "FreeBu helps me create Facebook friend lists". The second factor refers to the reflection and re-evaluation of one's friends in her EOSN. For example, "FreeBu clarifies my relationships with others of whom I am not fully aware". The regression analyses in [16] further identified several attributes that directed users' attention and guided users' usage of the tool. For example, in the map view of FreeBu, users are more interested in the friends who act like hubs (with high betweenness scores) or the friends who are outliers (with low degree scores) in their social networks. In the rank view, users were very interested in the friends to whom they often communicated.

6 Conclusion

In this section, we first summarize our research, then address the future work.

6.1 Summary

In this work, we addressed the issue related to privacy-decision-making in Online Social Networks (OSN). The available large amount of information about friends overwhelms a user. It is then difficult for the user to decide the audience for her online posts. Various research work has pointed to friend categorization. Indeed, the theories in categorical thinking and social networking limits provide us with further support. Leveraging humans' innate ability to process visual information, we developed an online visualization application to help users explore and group friends. It requires careful design choices in both visualizations and algorithms. We first reviewed various existing tools, identified their merits and limits. We then described our first tool based on the CircleTree visualization and the modularity-based community detection (MOD). The former is our new visualization design. We conducted a user study to investigate OSN users' visibility-decision performances with two different grouping methods, under the CircleTree visualization. The participants were divided into two groups, one used hierarchical, modularity-based community detection method (HMOD) interactively, the other used Facebook smart lists (FSL). We found that the former group of participants utilized the circles more efficiently the latter. This provides the evidence that HMOD is more supportive than FSL for visibility decisions. It also suggests that graph-based algorithms can produce the communities that match users' manual circles, more than attribute-based ones.

We then compared MOD with another community detection model, Generative Model for Friendships (GMF). It had been shown that GMF outperformed the other eight community detection models [39]. The corresponding nine algorithms were run on three ego-network datasets, and compared to ground-truth circles. We ran MOD and GMF on the sub-datasets (due to the availability of the data), and found that MOD outperformed GMF. We also examined the characteristics of the ground-truth circles and proposed the Multi-membership Modularity-based community detection method (MMOD) that produced overlapping communities, with similar overlapping rate to the ground-truth. We then found that MMOD outperformed MOD.

It is important to note that improving community detection algorithms alone is insufficient. Users need informative visualizations to comprehend her online friends and construct her own friend lists. Guided by relevant sociological research and visualization design taxonomies, we developed three more interactive visualizations that compensated the CircleTree visualization. The four visualizations are based on four different friend-grouping strategies. They incorporate similar list-construction user interfaces.

In summary, we started with the concerns for online privacy and contact management, studied users' behavior in visibility decision-making in online posting, compared different approaches for (semi-)automatic community detection, and eventually developed a full-fledged web application of interactive friend-exploration/grouping for Facebook users. We examined in detail the design choices from different perspectives: information visualization, community detection algo-

rithms, human cognition for visual perception and information processing, and social theory on social groups.

6.2 Limitations and Outlook

We identify several main improvements for FreeBu:

- In the four views, each friend is only represented by her name (the rank view also includes photos). More information, such as photo, profile, recent status and likes can be summarized in an "info box" that appears besides each focused friend.
- There often exist the friends who do not connect to other friends in an ego-network. The loners can be randomly mixed into the circles in the circle view or scattered in the force-directed graph layout in the map view. It is then more orderly to collect these loners into the same circle or to map them in proximity in the graph.
- We can improve the circle view by applying MMOD (Sect. 4.4).
- For the circle view, we notice that the user needs to zoom-in fairly deeply to reveal the friend names in a circle. This can be improved by modifying the label-revelation threshold. Also, the positioning of the name labels needs adjustment, so as to avoid overlaps, while maintaining a grouping structure.
- The graph layout in the map view can be colored according to the communities detected by MOD, similar to InMaps (Sect. 2). The drawback of discretizing community colors is that it ignores the continuity of the friend graph. Some friends are meant to be community-ambiguous. One way to address this issue is via gradient colors. First, a friend's membership to the circle is characterized some measure, e.g. its clustering coefficient (as that in Social Graph in Sect. 2). However, we need to be more careful to make people perceive such fusion as a natural transition between communities on the graph.
- Because zooming can create too much local focus and lose global context, it could be more helpful for users to add the "fisheye-view" [24] and "map-window" [6] functions in the circle and map views.
- In all the four views, we can add filtering and searching function to improve users' exploratory experience.
- FreeBu users have reported in some cases rendering visualizations is slow. Complex visualizations and user interactivity occupy a large part of browser resources, sometimes result slow response or crash. Though the current standard web technologies are encouragingly evolving, such as improved graphics rendering capabilities in HTML5, faster built-in Javascript engines, the browser-based computation power is still limited for large-scale, online, interactive visualizations. For tools like FreeBu, visualization programs need to be more economic.

Acknowledgments We thank the Flemish Agency for Innovation through Science and Technology (IWT) and the Fonds Wetenschappelijk Onderzoek—Vlaanderen (FWO) for support through the projects SPION (grant number 100048) resp. Data Mining for Privacy in Social Networks (grant number G068611N).

References

1. Acquisti A, Gross R (2006) Imagined communities: awareness, information sharing, and privacy on the facebook. In: Privacy enhancing technologies. Springer, pp 36–58
2. Allport GW (1979) The nature of human prejudice. Basic Books, New York
3. Auber D, Mary P (2013) Tulip—an information visualization framework dedicated to the analysis and visualization of relational data. URL (Weblog): http://tulip.labri.fr/TulipDrupal/ (Download: 1129 2013)
4. Backstrom L, Kleinberg J (2014) Romantic partnerships and the dispersion of social ties: a network analysis of relationship status on facebook. In: Proceedings of the 17th ACM conference on computer supported cooperative work & social computing. ACM, pp 831–841
5. Bastian M, Heymann S, Jacomy M (2009) Gephi: an open source software for exploring and manipulating networks. In: International conference on weblogs and social media
6. Beard D, Walker J (1990) Navigational techniques to improve the display of large two-dimensional spaces. Behav Inf Technol 9(6):451–466
7. Bernstein MS, Bakshy E, Burke M, Karrer B (2013) Quantifying the invisible audience in social networks. In: ACM SIGCHI conference on human factors in computing systems (CHI 2013)
8. Bisson G, Blanch R (2012) Improving visualization of large hierarchical clustering. In: 2012 16th International conference on information visualisation (IV). IEEE, pp 220–228
9. Blackwell S (2010) How to live: a life of Montaigne in one question and twenty attempts at an answer. Random House, Manhattan
10. Boyd DM (2008) Taken out of context: American teen sociality in networked publics. ProQuest
11. Bruls M, Huizing K, Van Wijk JJ (2000) Squarified treemaps. In: Data visualization 2000. Springer, pp 33–42
12. Chen YW, Lin CJ (2006) Combining SVMs with various feature selection strategies. In: Feature extraction. Springer, pp 315–324
13. Collins AM, Quillian MR (1969) Retrieval time from semantic memory. J Verbal Learn Verbal Behav 8(2):240–247
14. Cowan N (2001) The magical number 4 in short-term memory: a reconsideration of mental storage capacity. Behav Brain Sci 24(1):87–114
15. De Wolf R, Pierson J (2013) Whos my audience again? Understanding audience management strategies for designing privacy management technologies. Telemat Inform
16. De Wolf R, Gao B, Berendt B, Pierson J (2015) Interactive grouping technology for social network sites: exploring users' perceived values and actual behaviours, submitted for publication, at the ACM conference on computer-supported cooperative work and social computing
17. Dunbar RI (1992) Neocortex size as a constraint on group size in primates. J Hum Evolution 22(6):469–493
18. Dunbar R, Sutcliffe A (2012) Social complexity and intelligence. The oxford handbook of comparative evolutionary psychology p 102
19. Fang L, Kim H, LeFevre K, Tami A (2010) A privacy recommendation wizard for users of social networking sites. In: Proceedings of the 17th ACM conference on Computer and communications security. ACM, pp 630–632
20. Field DJ, Hayes A, Hess RF (1993) Contour integration by the human visual system: evidence for a local association field. Vis Res 33(2):173–193
21. Fortunato S (2010) Community detection in graphs. Phys Rep 486(3):75–174
22. Fortunato S, Barthelemy M (2007) Resolution limit in community detection. Proc Natl Acad Sci 104(1):36–41
23. Fruchterman TM, Reingold EM (1991) Graph drawing by force-directed placement. Softw: Pract Exp 21(11):1129–1164
24. Furnas GW (1981) The fisheye view: a new look at structured files. Technical report, Bell Laboratories Technical Memorandum
25. Furnas GW (1986) Generalized fisheye views, vol 17. ACM

26. Gao B, Berendt B, Clarke D, Wolf RD, Peetz T, Pierson J, Sayaf R (2012) Interactive grouping of friends in OSN: towards online context management. In: ICDM workshops. IEEE Computer Society, pp 555–562
27. Ghoniem M, Fekete JD, Castagliola P (2004) A comparison of the readability of graphs using node-link and matrix-based representations. In: IEEE symposium on information visualization INFOVIS 2004. IEEE, pp 17–24
28. Green-Armytage P (2010) A colour alphabet and the limits of colour coding. JAIC-J Int Colour Assoc 5
29. Grivet S, Auber D, Domenger JP, Melancon G (2006) Bubble tree drawing algorithm. In: Computer vision and graphics. Springer, pp 633–641
30. Gürses S (2010) Multilateral privacy requirements analysis in online social network services. PhD thesis, Department of Compute Science, KU Leuven
31. Healey CG, Enns JT (1999) Large datasets at a glance: combining textures and colors in scientific visualization. IEEE Trans Vis Comput Graph 5(2):145–167
32. Heer J, Shneiderman B (2012) Interactive dynamics for visual analysis. Queue 10(2):30
33. Johnson B, Shneiderman B (1991) Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In: Proceedings IEEE conference on visualization, Visualization'91. IEEE, pp 284–291
34. Kilmer R, Kilmer WO (2014) Designing interiors. Wiley, New York
35. Lee C, Cunningham P (2013) Community detection: effective evaluation on large social networks. J Complex Netw p cnt012
36. Macrae CN, Bodenhausen GV (2000) Social cognition: thinking categorically about others. Annu Rev Psychol 51(1):93–120
37. Marwick AE, Boyd Dm (2011) I tweet honestly, I tweet passionately: Twitter users, context collapse, and the imagined audience. New Media Soc 13(1):114–133
38. Mazzia A, LeFevre K, Adar E (2012) The PViz comprehension tool for social network privacy settings. In: Proceedings of the eighth symposium on usable privacy and security. ACM, p 13
39. McAuley JJ, Leskovec J (2012) Learning to discover social circles in ego networks. In: NIPS, pp 548–556
40. Miller GA (1956) The magical number seven, plus or minus two: some limits on our capacity for processing information. Psychol Rev 63(2):81
41. Moore K, McElroy JC (2012) The influence of personality on facebook usage, wall postings, and regret. Comput Hum Behav 28(1):267–274
42. Moreira AA, Paula DR, Costa Filho RN, Andrade JS Jr (2006) Competitive cluster growth in complex networks. Phys Rev E 73(6):65–101
43. Munkres J (1957) Algorithms for the assignment and transportation problems. J Soc Ind Appl Math 5(1):32–38
44. Newman ME (2005) A measure of betweenness centrality based on random walks. Soc Netw 27(1):39–54
45. Newman ME (2006a) Finding community structure in networks using the eigenvectors of matrices. Phys Rev E 74(3):36–104
46. Newman ME (2006b) Modularity and community structure in networks. Proc Natl Acad Sci 103(23):8577–8582
47. Nguyen QV, Huang ML (2002) A space-optimized tree visualization. In: IEEE symposium on information visualization INFOVIS 2002. IEEE, pp 85–92
48. Raynes-Goldie K (2010) Aliases, creeping, and wall cleaning: understanding privacy in the age of facebook. First Monday 15(1)
49. Rodrigues EM, Milic-Frayling N, Smith M, Shneiderman B, Hansen D (2011) Group-in-a-box layout for multi-faceted analysis of communities. In: 2011 IEEE third international conference on privacy, security, risk and trust (passat) and 2011 IEEE third international conference on social computing (socialcom). IEEE, pp 354–361
50. Seo J, Shneiderman B (2002) Interactively exploring hierarchical clustering results [gene identification]. Computer 35(7):80–86

51. Shannon P, Markiel A, Ozier O, Baliga NS, Wang JT, Ramage D, Amin N, Schwikowski B, Ideker T (2003) Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome Res 13(11):2498–2504
52. Shiffrin RM, Schneider W (1977) Controlled and automatic human information processing: II. Perceptual learning, automatic attending and a general theory. Psychol Rev 84(2):127
53. Shneiderman B, Dunne C (2013) Interactive network exploration to derive insights: filtering, clustering, grouping, and simplification. In: Graph drawing. Springer, pp 2–18
54. Shneiderman B, Wattenberg M (2001) Ordered treemap layouts. In: Proceedings of the IEEE symposium on information visualization 2001, vol 73078
55. Stevens SS (1957) On the psychophysical law. Psychol Rev 64(3):153
56. Sutcliffe A, Dunbar R, Binder J, Arrow H (2012) Relationships and the social brain: integrating psychological and evolutionary perspectives. Br J Psychol 103(2):149–168
57. Ugander J, Karrer B, Backstrom L, Marlow C (2011) The anatomy of the facebook social graph. arXiv preprint arXiv:11114503
58. Van Wijk JJ, Van de Wetering H (1999) Cushion treemaps: visualization of hierarchical information. In: Proceedings. 1999 IEEE symposium on information visualization. 1999 (Info Vis'99). IEEE, pp 73–78
59. Wang L, Giesen J, McDonnell KT, Zolliker P, Mueller K (2008) Color design for illustrative visualization. IEEE Trans Vis Comput Graph 14(6):1739–1754
60. Wang Y, Norcie G, Komanduri S, Acquisti A, Leon PG, Cranor LF (2011) I regretted the minute i pressed share: a qualitative study of regrets on facebook. In: Proceedings of the seventh symposium on usable privacy and security. ACM, p 10
61. Ware C, Purchase H, Colpoys L, McGill M (2002) Cognitive measurements of graph aesthetics. Inf Vis 1(2):103–110
62. Wolfram S (2013) Data science of the facebook world. http://blog.stephenwolfram.com/2013/04/data-science-of-the-facebook-world/, Retrieved 30 Nov 2013
63. Yi JS, ah Kang Y, Stasko JT (2007) Toward a deeper understanding of the role of interaction in information visualization. IEEE Trans Vis Comput Graph 13(6):1224–1231
64. Zuckerberg M (2012) One billion people on facebook. http://newsroom.fb.com/news/2012/10/one-billion-people-on-facebook/, Retrieved 19 Jul 2014

Genetically Optimized Realistic Social Network Topology Inspired by Facebook

Alexandru Topirceanu, Mihai Udrescu and Mircea Vladutiu

Abstract Social network analysis is receiving an increased interest from multiple fields of science since more and more natural and synthetic networks are found to share similar features which help us understand their underlying topological properties. One desire is to create a model of the human society, however, the complexity of such a model is increased by the nature of human interaction, and present studies fail to create a fully realistic model of the societies we live in. Our approach is inspired from studies of online social networking and the ability of genetic algorithms (GA) to optimize topological data in a natural manner. We combine the properties of the small-world and scale-free models to create a community-based social network, which is then rearranged using empirically obtained data from Facebook friendship networks, and optimized using GAs. As a result, our synthetically generated social network topologies are more realistic, with a proposed realism fidelity metric that is with 63 % closer to the observed real-world parameters than the best existing model.

Keywords Social networks · Real-life network topology · Genetic algorithms

1 Introduction

The effort to mathematically model an accurate and realistic society has been triggered by the observation of the three fundamental properties of social networks: average path length, clustering coefficient and degree distribution [1]. The well-known models of small-world [2] and scale-free [3] networks both present these

A. Topirceanu (✉) · M. Udrescu · M. Vladutiu
Department of Computers and Information Technology,
Politehnica University Timisoara, Bd. Vasile Parvan 2, 300223 Timisoara, Romania
e-mail: alext@cs.upt.ro

M. Udrescu
e-mail: mudrescu@cs.upt.ro

M. Vladutiu
e-mail: mvlad@cs.upt.ro

J. Kawash (ed.), *Online Social Media Analysis and Visualization*,
Lecture Notes in Social Networks, DOI 10.1007/978-3-319-13590-8_8

network properties but they fail in creating fully realistic models of the societies we live in. Over the years, many attempts have been made to merge as many empirically-observed properties as possible into a single social model. There are topological models which describe geographical proximity, friendship distribution, neural networks in the brain, protein interaction mechanisms, natural food chains, the distribution of means of transportation, citation networks, sexual interaction patterns, the world wide web, power distribution networks, relationship of words in a language, interaction between ingredients in a recipe, the world markets [4–6] etc. However, an abstract and generic, yet flexible and realistic model that describes how people interconnect in society has not yet been described. The benefit of having such a model is the capability of simulating custom social scenarios of interest on very large virtual data sets. It can help medical science predict the spreading of diseases [6]; sociology and politics to understand the flow of information, opinion etc.; and combined with diffusion models, help predict the outcome of elections, polls, surveys [6]. As there is not enough real world data to research on, this can only be simulated if we have dependable, realistic social models [5]. If we only rely on mining after existing real data found in online databases it becomes hard to find topologies with a certain distribution of properties. For example, if one needs to simulate the behavior of a realistic social network with a certain centrality distribution of nodes, our proposed algorithm can create such a desired network on demand, at the same time assuring that it is as realistic as possible.

This paper tackles the problem of synthetically generating realistic social network topologies. Unlike existing social models like the small-world [2], scale-free model [7], cellular model [8], static geographic model, Watts-Strogatz model with degree distribution [9], introduction model, random encounter model etc., we propose a methodology based on creating realistic models inspired from accurate empirical data which are then optimized using genetic algorithms. Previous methods fail in creating models of the human society; therefore we have used empirically obtained data from Facebook friendship networks and have concluded that, although diverse in shape and size, all share common metrics. We optimize the metrics on our model until it reaches a desired state of realistic accuracy.

Our synthetic model proves to be more efficient by replicating all network metrics measured on the empirical data set. We quantify this efficiency through a proposed realism fidelity metric δ which measures the realism of any model. The algorithm is highly parametrized, flexible for multi-purpose social scenarios, and is also being integrated into Gephi [10], the leading tool in visualization and analysis of large networks.

The paper is organized as follows: in Sect. 2 we introduce the reference data used for validating our synthetic topology, Sect. 3 discusses the problem of recreating realistic social models by presenting the state of the art and the means for evaluating the realism of the recreated models. For the latter problem we define a metric which best captures the network properties in complex networks context, as well as outline what the limitations of other existing metrics are. Section 4 describes the proposed algorithm step by step and exemplifies its usage on a small scale scenario. In Sect. 5 we present results regarding the accuracy of our proposal using the introduced metric

and applying it to measure the fidelity of our model, and of other state of the art models, against the reference social model we consider as realistic. Finally we draw some conclusions and present the future work in Sect. 6.

2 The Reference Data Set

Our research is based on the empirical study of Facebook friendship graphs which have been extracted from Facebook using an application named *netvizz* [11]. The data set consists of 93 different friendship graphs of subjects aged 16–35, with sizes ranging from 177 to 1,030 nodes. Even though different at first glance, under a closer numerical analysis all measured metrics vary only slightly between the multitude of networks. Figure 1a shows such a friendship network extracted from Facebook. We have measured the basic network metrics: network size (nodes and edges), *average path length* (L), *clustering coefficient* (C) and *average degree* ($\langle k \rangle$), and also network *diameter*, *density* and *modularity* [4]. Additionally we have analyzed the distributions of the *degrees* ($P(k)$), *betweenness*, *closeness* and (eigenvector) *centrality* [4, 12].

Figure 2a highlights the narrowness of the convergence intervals for average path length, clustering coefficient, density and modularity as measured for all networks extracted with *netvizz*. This strengthens the argument that all realistic topologies have metrics which fall inside these thresholds. In Fig. 2b we also show the degree and centrality distributions for a representative friendship network. The chosen network (Fig. 1a) lies nearest to all average values for each metric. Despite intuition and the inherent diversity of humans, it is clear that the measured values pertain to a social pattern which seems to be found in the underlying nature of the human interaction model. We have evaluated these parameters because they characterize a

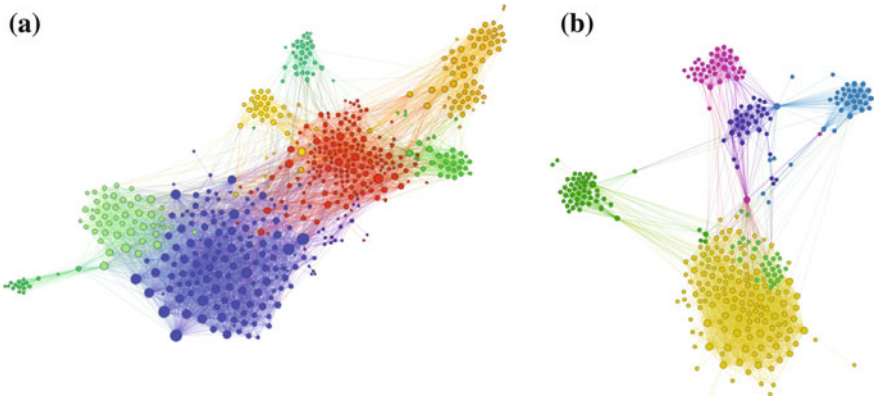


Fig. 1 Two online friendship networks: a Facebook network (**a**) of 590 nodes and a GooglePlus (**b**) network of 344 nodes. The size of each node is proportional to its degree and the coloring is done according to the community it belongs to (i.e. done by running a community finding algorithm first). This Facebook network is chosen as an example because it lies nearest to the overall average metric distribution from the data set

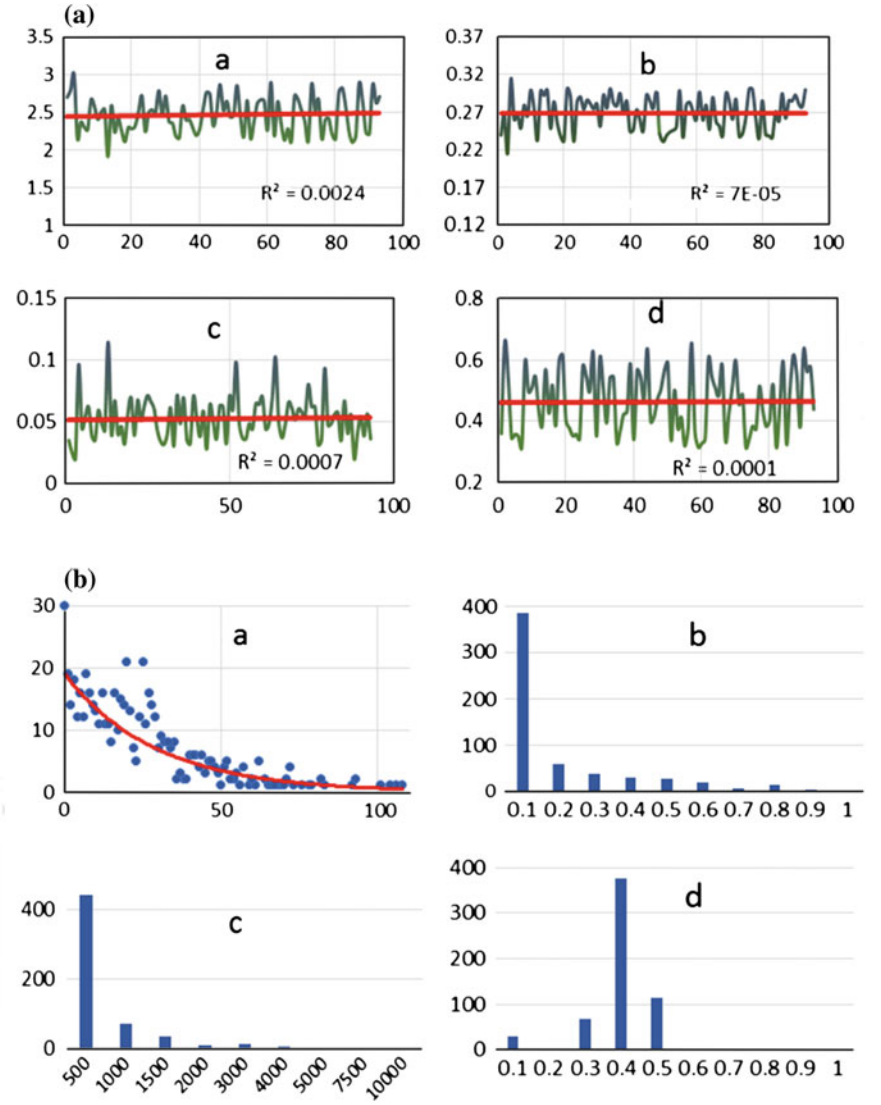


Fig. 2 **a** The distribution of measurements over the data set: (*a*) Average path length (L) with an aver-age value of 2.48, a minimum of 1.92 and a maximum of 3.0; (*b*) The clustering coefficient (C) with an average value of 0.26, a minimum of 0.21 and a maximum of 0.31; (*c*) Network density with an average value of 0.052, a minimum of 0.02 and a maximum of 0.11; (*d*) Network modularity with an average of 0.462, a minimum of 0.31 and a maximum of 0.65. Degree and centrality distributions for one representative network (represented in Fig. 1a). **b** The distributions for a representative network: (*a*) Power law degree distribution; (*b*) Power law eigenvector centrality distribution; (*c*) Power law betweenness centrality distribution; (*d*) Closeness centrality distribution. It presents a particular Gaussian distribution with a cutoff value (0.5). This is a specific feature for friendship networks

realistic social topology. Following similar reasoning, a study demonstrates that even a completely synthetic network—the Marvel characters universe—has evolved into a real-like social network [13]. Therefore the proposed synthetic topology generation process has to meet these demands.

The study presented in this paper is explicitly tailored to recreating Facebook friendship networks as we have concluded they best capture the aspects of social interconnectivity. Moreover, we rely on our private repository of social data which contains a large set of normal-sized Facebook user networks (~100–1,000 nodes), while other online resources, like the Stanford Large Network Dataset Collection (SNAP) [14], contain few very large non-representative networks for our study. As a future step, the study may be extended to extract the characteristics of Twitter, Google Plus, peer-to-peer networks (technological) and redefine the algorithm to suit additional specific cases.

3 Creating Realistic Societies

Empirical studies done over a variety of natural and man-made networks have resulted in the definition of several metrics used to describe and measure these networks. Focusing on the metrics explained and measured in Sect. 2, we present an overview of the recent related work in regard to social modeling, as well as highlight why each current social model does not meet the required accuracy. Also in this section, we present an overview and discuss the state of the art statistical methods used in networks comparison in general, and particularly, the ones applied in social network analysis.

3.1 Related Work

Current research to improve the accuracy of social topologies has been done by combining properties from the two fundamental models previously described with empirical data gathered from various contexts. A first notable study shows the impact of adding a power law degree distribution to small-worlds [9]. The *Watts-Strogatz model with degree distribution* (WSDD) is designed by creating a small-world topology (short L and high C) but also modifying the degree distribution of nodes, from a normal distribution to a power law one. *Cellular networks* have been proposed as a response to the need for large-scale multi-agent simulations [8]. They are based on the observation of covert networks, like the Al Qaeda terrorist organization. Cellular networks consist of an arbitrary number of normal-distributed sized cells, with a high clustering, in which a node is chosen as a cell leader. Further models exist that expand on the conclusions of Milgram's experiment [15]. The *static-geographic model* generates a social network in which links are added between nodes taking the actual distance into consideration: the greater the distance, the lower the wiring probability. The *introduction model* is similar to a small-world network, in the sense of recreating realistic triadic closures. Once a wiring is done between two nodes

with probability p_1, one node tries to connect to as many friends of the other node as possible using probability p_2. The *random encounter model* is useful for modeling population dynamics. Each node receives a random 2D movement and connects with a probability p to any other node it collides with. *Growth models* are variations of the Albert-Barabasi [7] algorithm and model realistic network growth according to the "rich get richer" principle. Such a model is the *WIW* online platform started as an experiment in 2002 in Hungary [16]. Analyzing the edges between over 45,000 users, the study proves the existence of a high clustering in real social networks and the fundamental role of triadic closures in creating new friendships.

Similar work based on friendship formation proposes the creation of a synthetic network to be used to simulate social interactions in a population for a given geographic space [17]. It predicts social travel focusing on friendship relationships and the results indicate that the model is able to generate networks that display the same structural properties as in the sample data.

3.2 Evaluating the Related Work

Basic network analysis is done by measuring fundamental graph metrics, and comparison of two or more networks, by doing an individual comparison of each metric independently. While such an approach is useful in trying to capture one specific feature of the network, it fails to create a general overview of the similarity between the analyzed networks [18]. Similar work aimed at comparing the importance of graph metrics concludes that each metric captures specific attributes of the network [19]. It states that further study on the effect of each metric is needed in order to be able to choose a fitting topology according to the requirements of an empirical model.

Comparing real systems is aimed at a deeper understanding of the interaction patterns between these systems [1–3], and extracting their common properties helps improve the models even further [2, 20, 21]. However, the predominant method of graph metric comparison suffers from limited information [22]. Some notable means of comparison are the distance ratio measure [23], used to compare individual mental models, a comparison from the data analysis perspective [22] and the study of the self-similarity of complex networks (Song 2005). From a topological perspective there are studies done both in the direction of classifying social network models [21] and of structural pattern detection [24]. These methods however serve a higher level of meta-analysis rather than as measures of similarity.

The statistical methods with which network similarity can be measured are the cosine similarity [25], variance, covariance, Pearson correlation coefficient (PCC) [26], the Mahalanobis distance [27]. Other methods used in network analysis which are adopted from statistics include the T-test and the ANOVA test (analysis of variance). A recent study improves upon the T-test methodology by proposing an alternative geometrical approach called the Characteristic Direction, in order to identify differentially expressed genes [28]. There is no single statistical approach used in current research because there is no unified metric that provides normalized values which are tailored specifically to network comparison. Yet, the most intuitive

Table 1 Measurements for average degree (*AvgD*), average path length (*L*), average clustering coefficient (*C*), modularity (*Mod*), diameter (*Dmt*) and density (*Dns*) are done on synthetically generated networks of 1,000 nodes

	AvgD	*L*	*C*	*Mod*	*Dmt*	*Dns*	δ
Facebook	19.822	2.4815	0.2659	0.4677	8.5	0.0496	0
SW	3.994	5.614	0.321	0.726	11	0.005	0.371
SF	3.12	4.598	0.015	0.622	10	0.003	0.38
Cellular	11.388	3.786	0.599	0.908	7	0.02	0.367
Geographic	6.628	3.34	0.065	0.52	8	0.013	**0.277**
WSSD	21.583	4.589	0.738	0.897	9	0.041	**0.31**
Genosian	20.02	2.404	0.308	0.65	5	0.05	**0.125**

The presented models are: Facebook, small-world (SW), scale-free (SF), cellular, static-geographic, WSDD and the proposed model (Genosian). δ shows the realism of the models, with the geographic model being the most accurate state-of-the-art model (0.27), but with the Genosian (proposed) model being 122 % more realistic (0.125)

and thus used metrics are the Euclidean distance, Pearson correlation [29] and cosine similarity [25].

Analyzing the results from Table 1 and Fig. 3 we can highlight the limitations of the current related work. No single state of the art model manages to replicate more than two or three empirical measurements. As there is no existing metric to compare and quantify the realism of social networks, this paper makes use of the network fidelity metric δ (delta) which was previously introduced in literature by the authors [30]. By measuring the δ of any two networks represented with the same metrics, it can be concluded which model offers the greatest realism compared to our empirical data set. Also, the algorithms for generating all the analyzed networks, as originally described by their respective authors, are implemented as Gephi plug-ins by the authors.

$$sd_i = \begin{cases} 1 - \frac{\bar{m}_i}{2\bar{m}_i - \bar{m}_i'} & , \quad \text{if } \bar{m}_i' < \bar{m}_i \\ 1 - \frac{\bar{m}_i}{\bar{m}_i'} & , \quad \text{if } \bar{m}_i' \geq \bar{m}_i \end{cases} \tag{1}$$

$$\delta = \frac{1}{n} \sum_{i=1}^{n} sd_i \tag{2}$$

where $i = \{1, 2, \ldots, n\}$ is the index of the average metric which describes the two models being compared and n is the total number of metrics. $\bar{m}_i'$ is the measured metric (e.g. modularity) and $\bar{m}_i$ is the empirical average for that same metric (e.g. modularity of the Facebook data set). The closer the value of δ is to 0, the closer a model resembles the Facebook data set. The two branches assure symmetry.

Because none of the presented work manages to fully model a realistic friendship network, as modeled on Facebook, we further propose our own topological model which encompasses all the metrics and distributions.

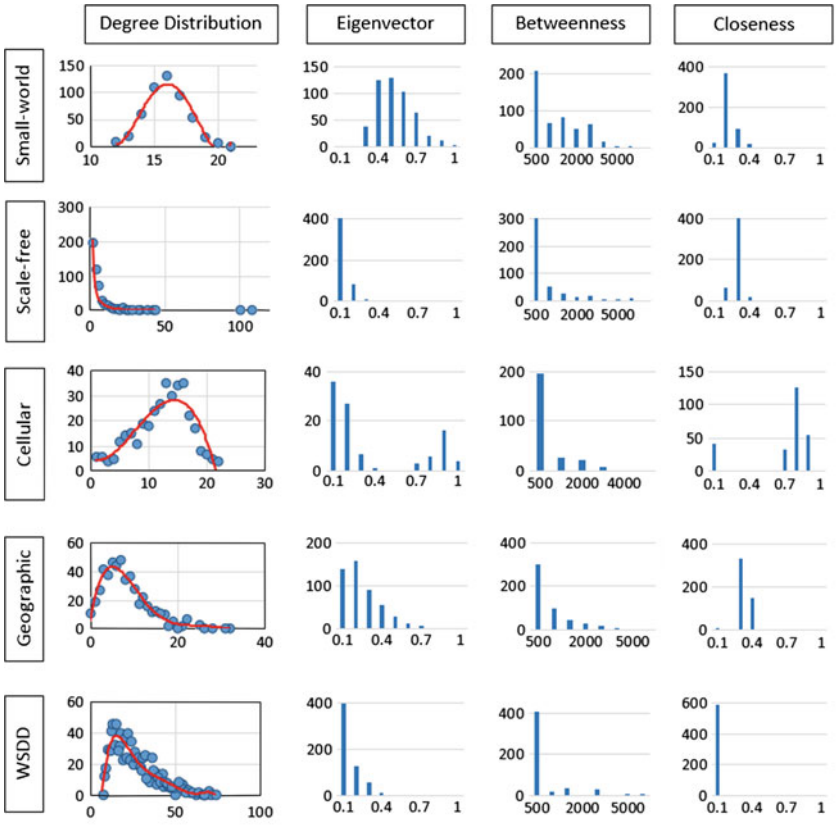


Fig. 3 The degree and centrality distributions over a selection of five relevant social network models (the same ones as described in Table 1)

4 The Genetic-Optimized Social Network (GenOSiaN)

The goal of our proposed social network model (called *Genosian*) is to create an accurate replica of the friendship models gathered from Facebook. Figure 4 presents the overview of our proposed methodology. As there is no direct algorithm of ensuring the creation of a graph with the desired predefined metrics we adopt a step wise refinement approach. By combining the two fundamental models, small-world and scale-free, a community based network is created. This approach is also validated by similar research, namely the cellular network model [8], the Watts-Strogatz model with degree distribution [9], and scalable virtual communities [31]. The resulting communities are connected with initially random friendship links which are then re-wired (optimized) using a genetic algorithm (GA) approach. Related work shows that reciprocity and sibling bias are shown to have a considerable effect over the creation of friendship ties [32], thus our friendships tend to form around the dominant

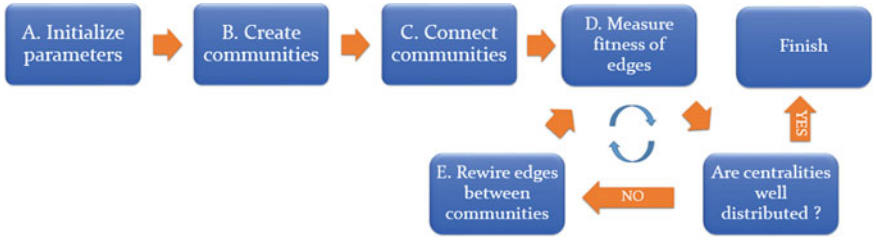


Fig. 4 Flowchart describing the steps of the Genosian algorithm

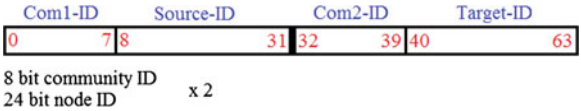


Fig. 5 The genetic chromosome representation. Each solution is composed out of two 32bit-represented IDs. The source node is represented by concatenating the community ID of the node (8bit) with the actual node ID (24bit). The same rule applies for the edge target

nodes in each community. Using a custom GA the centrality distributions are repeatedly measured and optimized until they correspond to the empirical distributions.

To overcome the problem of searching in an infinite solution space we propose the usage of heuristic methods, namely genetic algorithms (GAs), to solve this computational limitation. GAs are used to generate a representative mixture of solutions to optimization and search problems [33].

The initial set of random solutions which build up the population is, in this particular case, composed out of edges (candidate solutions) which need repeated rewiring, as in a genetic manner, to produce better and better solutions. Finally, after a predefined number of repetitions (generations) the GA stops and the best solution is chosen from the population [33]. Solutions are ordered using a fitness function, namely the betweenness and/or the eigenvector centrality of each edge target. Figure 5 shows the chromosome representation used by the algorithm. Each solution (candidate) consists of en edge: a pair node source—node target on which the genetic operators are applied.

4.1 Initialize Parameters

The algorithm takes as input the following: the number of communities, the average community size, the two rewiring probabilities p_1 and p_2, and the genetic algorithm parameters. Each community is individually built using the same principle, so the number of communities is used to model diverse real situations, like e.g. number of college groups. By default, the average community size determines a power law distributed size around the given value; alternatively, each communities' size can be manually set.

4.2 Create Communities

The creation of each community is an independent task and is inspired by the Watts-Strogatz algorithm. The proposed difference consists in how the value k is individually chosen for each node, as inspired from the WSDD approach. It is aimed at creating a small world network with a power law degree distribution. After the regular network ring is created, local edges are rewired to long range edges, within the community, using probability p_1. At this step we obtain a given number of communities, each with realistic L, C, density and degree distribution.

```
Create community:
create size nodes with id = community index | global node index
for each node n_i:
  connect n_i to k neighbors on left and right sides (2*k edges)
for each edge e_i, with probability p_1:
  choose a new random/preferential edge target from the community
```

4.3 Connect Communities

The last initialization step consists of connecting the obtained communities. Using probability p_2 each node is connected with another random or preferentially chosen node from a different community. The preferential selection is done by choosing higher degree nodes in favor of lower degree ones. This step stabilizes the graph density, diameter and modularity, but the centralities remain normally distributed. After this step, the list of added inter-community edges is kept for the next iterations of the algorithm.

```
Connect communities:
for each community c_a:
  for each node n_i from c_a, with probability p_2:
    choose another random community c_b
    choose a random/preferential node n_j from c_b
    create edge e_k between (n_i, n_j) and save it to list E
```

4.4 Measure Fitness of Edges

The list of newly created edges is sorted in descending order of the betweenness and/or Eigenvector centrality of the target of each edge. For this we run the corresponding centrality measurement algorithms and then order the edges. The idea is to rewire the edges by keeping the source node, but selecting a better target node. Better targets represent nodes with higher centrality, as our empirical observations suggest. The GA can be repeated for either a given number of steps (iterations) or until the measured centralities resemble the empirical ones. Experiments show that running a higher number of iterations (>5) makes the network organize itself in a

perfect manner, which actually decreases the realistic accuracy. Consequently, we suggest using the algorithm with a fixed number of steps (2–5).

```
Measure fitness:
for each edge e_i in E:
  fitness f_i ← centrality(target n_i of e_i)
sort E in descending order of f_i
```

4.5 Rewire Edges Between Communities

Considering we sort the population in *descending* order of the fitness after every iteration, we evolve the solutions from one generation to the next using three methods:

1. *Best solutions*: the first percentage *pBest* of the current generation is copied to the next generation. That is, the edges with the most central target nodes (best fitness) are kept in the graph (there are $sBest = pBest \times N$ individuals chosen).
2. *Crossover*: a second percentage *pCross* of the next generation is composed out of edges from the current population on which a custom crossover is applied (there are $sCross = pCross \times N$ individuals chosen for crossover). It is applied on the edge target using a second random target node chosen from the same community. The local IDs from the chromosome of the original target and the second random target are combined through binary concatenation using the first c bits from one node's ID and the remaining bits from the other node's ID. The crossover threshold c is a random number: $0 \leq c \leq n$, where n is the number of bits used to represent the IDs. The resulting index is guaranteed to belong to a specific node in the community, which is then set as the new target for the particular edge on which the crossover is applied.
3. *Mutation*: a third percentage *pMutation* is composed out of the remaining edges from the current population on which genetic mutation is applied (there are $sMutation = pMutation \times N = N - sBest - sCross$ individuals chosen fro mutation). It is applied by changing the edge target node with another random or preferential node from the same community as the target, as given by the fitness function.

The three percentages add up to 100 %. Finding the best values for them is an experimental study and may differ from one need to another. Figure 6 explains how the algorithm is applied step by step.

```
Apply genetic operators (one step):
E' ← empty
while i++ < sBest:
  E'_i ← E_i
while i++ < sBest + sCross:
  n_j ← random node from community of target of (E_j)
  new_idx ← crossover (getNodeId(n_i), getNodeId(n_j))
  n_new ← getNode (new_idx) from same community
  target (E_i) ← n_new, add to E_i
```

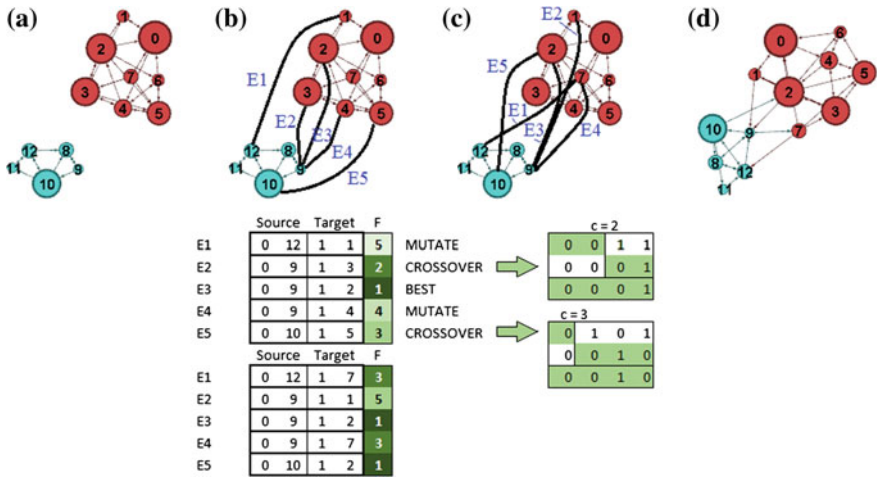


Fig. 6 Step by step explanation of the *Genosian* algorithm. The table shows the evolution of the chromosomes, their fitness ranking (*green*), and exemplifies the crossover. **a** Communities C0 (*cyan*) and C1 (*red*) are created [steps A, B of the algorithm]. **b** Five random edges are drawn be-tween the two communities: E1 to E5 [step C of the algorithm]. The fitness F of the edges is computed and the edges are ordered (1–5 in the *green column*). F is given by the centrality of each edge's target, i.e. which is proportional to the size of the nodes in the figure [step D of the algorithm]. **c** Apply the genetic operators and rewire the five edges. Thus, in order of the fitness, the best solution is copied over (E3), crossover is applied on the next two solutions (E2, E5), and mutation on the last two solutions (E4, E1). Crossover on E2 is applied by combining the target "3" with a random target "1", with $c = 1$, which results in the new target "1". Mutation on E1 is applied by choosing a new random target from the same community, namely node "7" [step E of the algorithm]. **d** Considering the algorithm is finished after one step, the ForceAtlas2 layout algorithm is reapplied on the graph [34]

```
while i++ < size(E):
  n_r ← target of random/preferential edge from E
  target (E_i) ← n_r, add to E_i
E ← E'
```

Once the algorithm stops, it produces a graph of size $N \simeq$ *number of communities* $\times$ *average community size*, which significantly resembles the presented Facebook friendship networks. The basic metrics (average degree, L, C, diameter, density, and modularity) are realistically obtained through steps A-C of the algorithm, while steps D-E ensure the centralities are distributed as needed. Finally, in Fig. 6 we exemplify the algorithm in a step-by-step manner and explain how the rewiring is done.

5 Results

In this section we present and compare the similarities between the Genosian network, the real Facebook data set, and the best two models presented in the related work chapter, both visually (Fig. 7) and numerically (Table 2). Comparing the results in

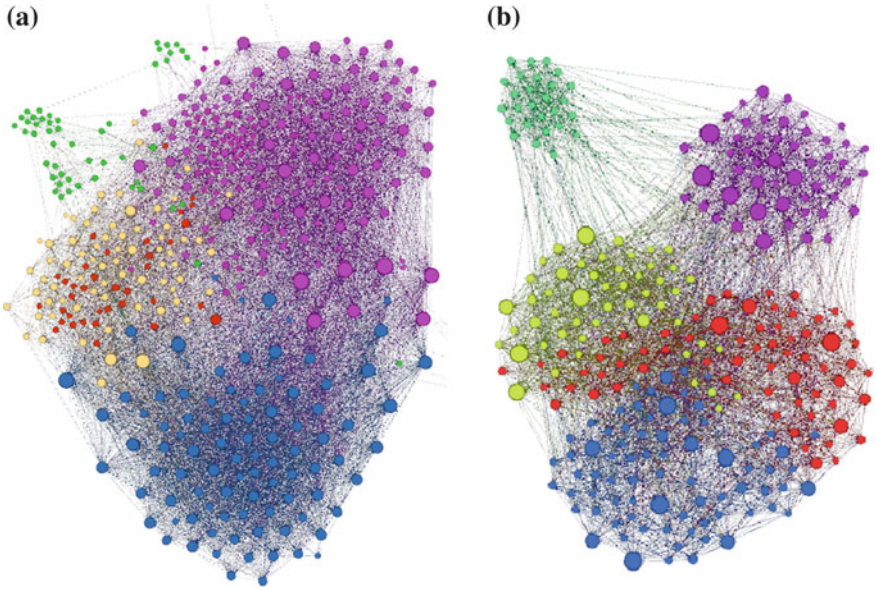


Fig. 7 A visual comparison between a Facebook friendship network (**a**) with 457 nodes, and a Genosian network (**b**) with 269 nodes. The synthetic network in the figure is the one corresponding to G2 in Table 2. The coloring of all nodes is done according to the community they belong to, and their size is proportional to their degree

Table 2 The basic metrics for a representative Facebook friendship network and for six Genosian networks of sizes 500–1,000 nodes

	AvgD	*L*	*C*	*Mod*	*Dmt*	*Dns*	p_1	p_2	δ
Facebook	19.82	2.481	0.266	0.47	8.5	0.05	–	–	0
G1	22.26	2.514	0.343	0.62	4	0.032	0.1	0.1	0.2
G2	22.17	2.426	0.212	0.5	4	0.029	0.1	0.2	0.166
G3	21.18	2.185	0.142	0.36	3	0.046	0.1	0.3	1.89
G4	21.45	2.416	0.324	0.65	4	0.044	0.2	0.1	0.167
G5	21.88	2.353	0.182	0.49	4	0.033	0.2	0.2	0.172
G6	20.02	2.404	0.308	0.65	5	0.05	0.3	0.1	**0.125**

The two columns on the right represent the wiring probabilities p_1 and p_2 (see steps A, B, C of the algorithm) used to create each of the distinct synthetic networks. The values used for the genetic percentages are: $pBest = 50\,\%$, $pCross = 30\,\%$, $pMutation = 20\,\%$

Table 2 with the ones in Table 1, it is clear that the Genosian networks manage to accurately replicate the original Facebook network. No other model reproduces more than 3 basic metrics ($0.27 < \delta < 0.38$), while the proposed model reproduces 5 out of 6 ($0.125 < \delta < 0.2$). Thus our model is, on average, 63 % more accurate than the best previous model (Geographic); the best network is however 2.21 times more accurate than the Geographic model, and 2.47 times more accurate than the

WSDD model. It is worth mentioning that our synthetic networks do not create the random leaf nodes which increase the diameter, as they are statistically insignificant, thus the real diameter is lower. The six networks highlight the impact of the wiring probabilities. A low p_2 creates a very modular community structure which is not desired. Increasing p_2 decreases the clustering and the modularity, and increasing p_1 increases the density. Figure 7, along with Fig. 1, highlight the results in a visual manner. One can observe the similarity between the Facebook friendship networks (Figs. 1a and 7a) and the proposed Genosian network (Fig. 7b). Nevertheless, the numerical comparison is the one that makes us conclude how realistic a model is, and both Tables 1 and 2 reinforce the fact that our proposal produces the best reproduction of real friendship network topologies. In Fig. 8 we also show the distributions of centralities for the Genosian model. This further stresses the superior realism of our proposed model because, in comparison with the other models, it represents a better match for the actual Facebook distributions from Fig. 2. Throughout the paper we have used Facebook friendship networks as a basis for comparison and validation of realism. This is argumented by the popularity of Facebook, which offers large, diverse and real-like networks, as explained in Sect. 2. However, other platforms, like GooglePlus (Fig. 1b), Twitter, Wikipedia etc., offer empirical data for validation. In principle, there is no problem for Genosian in replicating topologies that resemble GooglePlus or Twitter, provided that similar studies, as those pertaining for Facebook (as provided in Sect. 2), are performed.

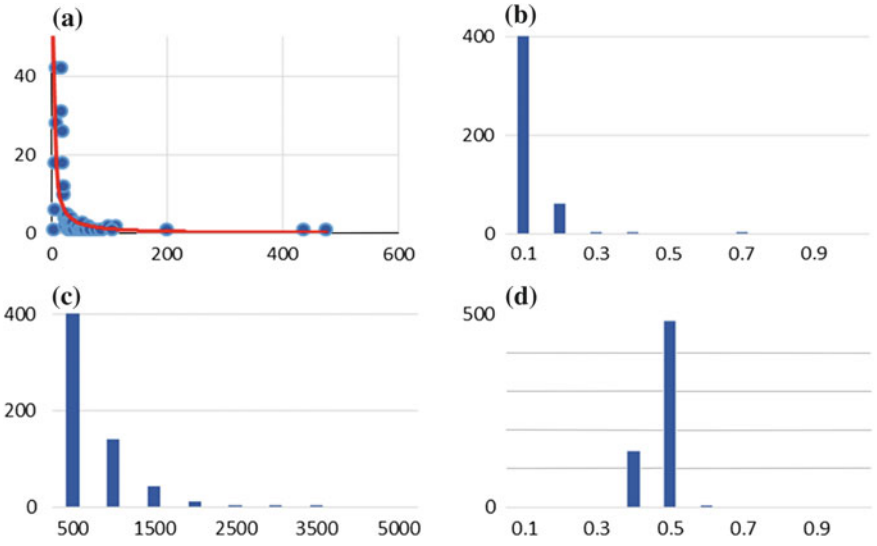


Fig. 8 The centrality distributions for a representative Genosian network. See Fig. 2b for comparison. **a** Power law degree distribution. **b** Power law Eigenvector centrality distribution, exactly as in Facebook networks, and unlike many other social network models. **c** Power law betweenness distribution. **d** Closeness distribution with the same particular Gaussian distribution as in real Facebook topologies

6 Conclusions and Future Work

Modeling the societies we live in is one of the goals of social network analysis. This endeavor stretches out in multiple directions: defining a mathematical model of the topology, modeling real-time growth, adding an opinion diffusion model etc. Many natural and synthetic networks have already been analyzed and their underlying models understood, documented and reproduced. However, the quest remains open to propose an accurate model of the society. While it encompasses the most fundamental properties—small-world and scale-free—it also brings a lot of complexity due to the nature of human interaction. Our proposal—the Genosian network—is an innovative solution combining a realistic empirical data set with social network analysis and genetic algorithm optimization.

The paper begins by showing that Facebook friendship networks are accurate at reproducing the real friendship networks between humans. The data set of 93 such networks, with sizes ranging from 177 to 1,030 nodes, is then analyzed and it is concluded that, although very diverse in shape and size, all these networks share very strict metrics. Further, we present the work related to newer proposals in this direction. We show that none of these manage to replicate the properties of the empirical friendship networks. Thus, we propose the Genosian network model and explain how it is algorithmically generated. We finalize by offering both a visual and a numerical comparison and discussion between the proposed, empirical and other related models.

We can conclude that our work has achieved its goal and manages to replicate realistic societies very accurately. The achieved accuracy shows a 63 % improvement with respect to the best previous model (static-geographic model), in terms of the realism fidelity metric δ. The inspiration from Facebook is nothing but natural, as more and more virtual data is modeled by human interaction. Our future work is aimed at extending the real social network topology study for Google Plus, Twitter, etc. in order to refine the Genosian algorithm accordingly. We also plan on using this model as a basis for social network growth.

Acknowledgments This work was partially supported by the strategic grant POSDRU/159/1.5/S/137070 (2014) of the Ministry of National Education, Romania, co-financed by the European Social Fund—Investing in People, within the Sectoral Operational Programme Human Resources Development 2007–2013.

References

1. Strogatz SH (2001) Exploring complex networks. Nature 410(6825):268–276
2. Watts DJ, Strogatz SH (1998) Collective dynamics of small-world networks. Nature 393(6684):440–442
3. Barabási A-L, Albert R (1999) Emergence of scaling in random networks. Science 286(5439):509–512
4. Wang XF, Chen G (2003) Complex networks: small-world, scale-free and beyond. IEEE Circuits Syst Mag 3(1):6–20

5. Hoffmann AOI, Jager W, Von Eije JH (2007) Social simulation of stock markets: taking it to the next level. J Artif Soc Soc Simul 10(2)
6. Easley D, Kleinberg J (2010) Networks, crowds, and markets, vol 8. Cambridge University Press, Cambridge
7. Albert R, Barabási A-L (2002) Statistical mechanics of complex networks. Rev Mod Phys 74(1):47
8. Tsvetovat M, Carley KM (2005) Generation of realistic social network datasets for testing of analysis and simulation tools. DTIC Document, Technical report
9. Chen Y, Zhang L, Huang J (2007) The Watts-Strogatz network model developed by including degree distribution: theory and computer simulation. J Phys A: Math Theor 40(29):8237
10. Bastian M, Heymann S, Jacomy M (2009) Gephi: an open source software for exploring and manipulating networks. In: ICWSM
11. Rieder B (2013) Studying facebook via data extraction: the Netvizz application. In: Proceedings of the 5th annual ACM web science conference. ACM, pp 346–355
12. Kunegis J, Fay D, Bauckhage C (2010) Network growth and the spectral evolution model. In: Proceedings of the 19th ACM international conference on information and knowledge management. ACM, pp 739–748
13. Alberich R, Miro-Julia J, Rosselló F (2002) Marvel universe looks almost like a real social network. ArXiv preprint cond-mat/0202174
14. Leskovec J (2011) Stanford large network dataset collection. http://snap.stanford.edu/data/index.html
15. Milgram S (1967) The small world problem. Psychol Today 2(1):60–67
16. Csányi G, Szendrői B (2004) Structure of a large social network. Phys Rev E 69(3):036131
17. Arentze T, van den Berg P, Timmermans H (2012) Modeling social networks in geographic space: approach and empirical application. Environ Plan—Part A 44(5):1101
18. Costa LdF, Rodrigues FA, Travieso G, Villas Boas P (2007) Characterization of complex networks: a survey of measurements. Adv Phys 56(1):167–242
19. Bigdeli A, Tizghadam A, Leon-Garcia A (2009) Comparison of network criticality, algebraic connectivity, and other graph metrics. In: Proceedings of the 1st annual workshop on simplifying complex network for practitioners. ACM, p 4
20. Amaral LAN, Scala A, Barthélémy M, Stanley HE (2000) Classes of small-world networks. Proc Natl Acad Sci 97(21):11 149–11 152
21. Kantarci B, Labatut V (2013) Classification of complex networks based on topological properties. In: Proceedings of the 3rd international conference on social computing and its applications
22. Li W, Yang J-Y (2009) Comparing networks from a data analysis perspective. Complex sciences. Springer, Berlin, pp 1907–1916
23. Caseiro N, Trigo P (2012) Comparing complex networks: an application to emergency managers mental models
24. Park K, Han Y, Lee Y-K (2013) An efficient method for computing similarity between frequent subgraphs. In: Proceedings of the 3rd international conference on social computing and its applications
25. Tan P-N et al (2007) Introduction to data mining. Pearson Education India, Upper Saddle River
26. Stigler SM (1989) Francis Galton's account of the invention of correlation. Stat Sci 4(2):73–79
27. Mahalanobis PC (1936) On the generalized distance in statistics. Proc Natl Inst Sci (Calcutta) 2:49–55
28. Clark NR, Hu K, Chen EY, Duan Q, Maayan A (2013) Characteristic direction approach to identify differentially expressed genes. ArXiv preprint arXiv:1307.8366
29. Spertus E, Sahami M, Buyukkokten O (2005) Evaluating similarity measures: a large-scale study in the Orkut social network. In: Proceedings of the eleventh ACM SIGKDD international conference on knowledge discovery in data mining. ACM, pp 678–684
30. Topirceanu A, Udrescu M, Vladutiu M (2013) Network fidelity: a metric to quantify the similarity and realism of complex networks. In: Proceedings of the 3rd international conference on social computing and its applications

31. Rana OF, Akram A, Lynden SJ (2005) Building scalable virtual communities: infrastructure requirements and computational costs. In: Socionics. Springer, Berlin, pp 68–83
32. Nakada T, Kato Y, Kunifuji S (2007) A study on the dynamics of friendship network formation using a directed network model
33. Mitchell M (1998) An introduction to genetic algorithms. MIT Press, Cambridge
34. Jacomy M, Heymann S, Venturini T, Bastian M (2011) Forceatlas2, a continuous graph layout algorithm for handy network visualization. Medialab Center of Research

A Workbench for Visual Design of Executable and Re-usable Network Analysis Workflows

Tilman Göhnert, Andreas Harrer, Tobias Hecking and H. Ulrich Hoppe

Abstract In this paper we introduce the concept of a web-based analytics workbench to support researchers of social networks in their analytic processes. Making explicit these processes allows for sound design, re-use, and automated execution using an authoring system for visual representations of these analytic workflows. The workbench is implemented according to a flexible technical framework in which external and newly-defined analytic components can be integrated and used in conjunction with other analytic components. As a showcase we discuss the integration of a complex analytic process using multi-relational blockmodeling on a well-known reference data set.

Keywords SNA research tool · Analysis workflows · Multi-relational blockmodeling

1 Introduction

Analyzing social networks is often a process comprised of multiple steps. Common network analysis tools like Pajek [1] or the Network Workbench [2] allow performing many of these steps in one single tool. The Network Workbench also offers a plugin mechanism which allows integrating additional analysis techniques into the existing tool without the need for creating a new one or switching between tools. However none of these tools offer a visual representation of complex

T. Göhnert (✉) · T. Hecking · H.U. Hoppe
University Duisburg-Essen, Lotharstr. 65, 47057 Duisburg, Germany
e-mail: goehnert@collide.info

T. Hecking
e-mail: hecking@collide.info

H.U. Hoppe
e-mail: hoppe@collide.info

A. Harrer
Clausthal Technical University, Julius-Albert-Str. 4, 38678 Clausthal-Zellerfeld, Germany
e-mail: andreas.harrer@tu-clausthal.de

J. Kawash (ed.), *Online Social Media Analysis and Visualization*,
Lecture Notes in Social Networks, DOI 10.1007/978-3-319-13590-8_9

multi-step analysis processes. Such a visual representation can be useful as a means for discussing and describing analysis processes. Additionally the explicit representation of analysis workflows can be used for re-executing and thus re-using a complete workflow. In combination with the opportunity to later modify a given workflow, the explicit representation also allows performing of multiple similar but not fully equal analysis processes without the need for building the complete workflow again each time it should be executed.

In this paper we introduce in Sect. 2 a new network analysis tool, the Analytics Workbench, which offers both a visual representation of analysis workflows and an extension mechanism which allows for simple inclusion of additional analysis techniques into the workbench. The integration and implementation aspects are described in Sect. 3. To demonstrate the flexibility of our approach and re-usability of analysis processes we use in Sect. 4 a recently developed technique for blockmodel analysis on multi-relational networks as a showcase for both the extension mechanism of the Analytics Workbench and how the visual representation can be used to make complex analysis processes explicit. Section 5 discusses the benefits of the workbench approach and gives details about a recently conducted evaluation with expert users. In Sect. 6 we summarize our results and give directions for our planned future work.

2 Overall Concept of the Analytics Workbench and Related Work

The development of the Analytics Workbench presented in this paper was motivated by two main factors which also determine the main goals of the development. One of these factors was the wish for an analysis tool which allows easy accessibility, allows complex analysis processes in an integrated environment, and offers an explicit representation of analysis workflows which can be used for sharing these workflows. The other factor was the need for an analysis tool which offers mechanisms that allow an easy integration of additional analysis functionalities, for example for the prototypical implementation of newly developed analysis approaches.

The goals easy accessibility and explicit representation of analysis workflows however mainly affect the user interface of the Analytics Workbench. The wish for easy accessibility also included the idea of a tool that does not need local installation on the users machine but can offer its capabilities through the internet, in the sense of cloud-computing and software-as-a-service [3]. For making the analysis processes or analysis workflows explicit we decided to make use of a graphical language that allows a visual representation of each analysis step as well as of the data flow, oriented on the pipes-and-filters design pattern [4]. There are several examples for such languages, including the activity diagrams of the Unified Modeling Language (UML) [5] or Petri Nets [6]. A straightforward approach is to visualize every single step of the analysis process as one graphical element or *module* and make the flow of data explicit by connecting the modules with links or arrows. This approach is used

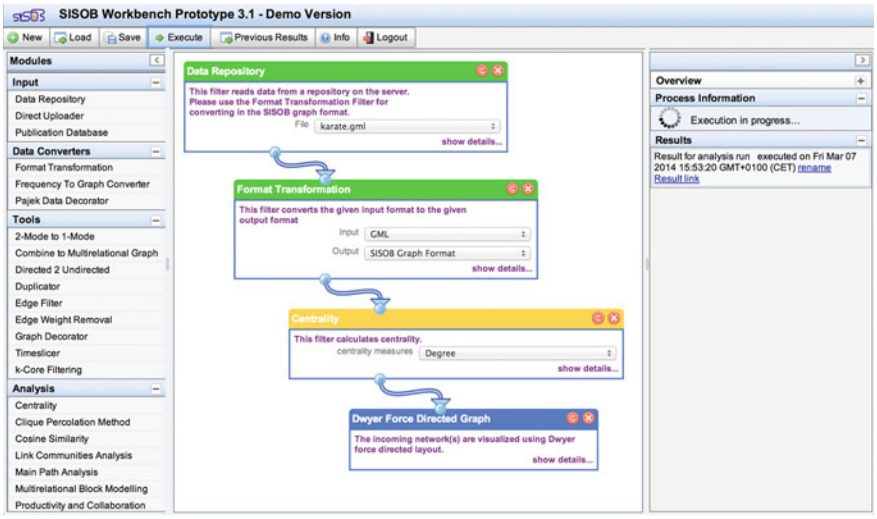


Fig. 1 User interface of the Analytics Workbench

in analysis tools like RapidMiner [7] or Yahoo Pipes [8] and is also the approach we chose for the Analytics Workbench. See Fig. 1 for a screenshot of the user interface showing a simple four-step network analysis workflow. These workflow descriptions can both be stored on the workbench server and also shared with other users of the workbench. Additionally for each workflow execution the workflow which was used is stored and linked to the results created in this workflow run, which are also stored on the server. Thus it is always possible to load not only results created earlier but also the workflow that was used to create them. We have combined this web-based user interface with a computational backend that performs the analysis processes described in the workflow descriptions and which will be the main focus of this paper.

Looking at the two main goals behind the development of the Analytics Workbench, there is already a number of tools and approaches for different application domains available that are intended to fulfill these needs. However none of them fully matched our goals. Two examples for such tools and approaches are the diviz tool [9] and the CIShell framework [10].

Diviz is a tool for designing, executing and sharing workflows from the field of Multiple Criteria Decision Aid (MCDA). Many of the design decision behind the development of diviz are similar to our approach. The user interface also makes use of a pipes-and-filters metaphor in which filters represent the individual steps of a workflow and connections between these filters define the data flow. The user interface is implemented in a proprietary platform-dependent client, which connects to web-services that are used for performing the actual analysis calculations. This approach for dividing labor between a client that mainly acts as user interface and server based execution is similar to the one we have implemented in the Analytics

Workbench. However we wanted to have a web-based user interface as that would cover an even wider range of platforms and makes the installation of an individual client obsolete.

CIShell itself is not a tool but a framework for so-called cyberinfrastructures. It has by now been used to implement a number of different tools, of which the Network Workbench (NWB) [2] is one example that is related to network analysis. CIShell uses the Open Services Gateway Initiative (OSGi) specification as basis for a framework in which plugins in the form of services can be used to add elements to a system. These services can provide data, algorithms, or other services to the tool. The general idea is that for creating a specific tool (like the NWB), a number of services or plugins are bundled to form the desired tool. In terms of extensibility and flexibility the approach of the CIShell is similar to the approach of the Analytics Workbench, as both allow adding functionality by implementing small components that adhere to well defined interfaces for the communication with the system they should be connected to. However there are two main differences. Although the CIShell approach in principle would also allow splitting user interface and execution of analysis processes, the currently available tools based on this framework are stand-alone tools to be installed locally on the user's computer. The other main difference is the user interface. The user interface of the Analytics Workbench is centered around the explicit workflow representation and thus facilitates seeing the workflow as a whole and also sharing workflows, as these can be stored on the workbench server with the option to make them available also to other users. Again the CIShell framework would allow building such a user interface, however the default interaction with tools implemented using CIShell is based on selecting individual actions to perform. There is a built-in scripting facility, which would also allow in a way storing and sharing the process, but it is not such a central element of the interaction with the system.

3 Flexible Integration Approach and Implementation

As stated above, one of the main goals behind the development of the Analytics Workbench was to allow easy integration of additional components, be it for customizing an installation of it for a certain analysis purpose or for simple addition of new analysis approaches. In order to do that, we decided to use a multi-agent approach, because this supports well the loose coupling of different analysis and visualization components, the extensibility and potential distribution of the system for rapid testing of new components. Since some programming languages and paradigms are especially suited for specific analyses, e.g. readily available libraries for natural language processing in Python to support word network analysis, a language-transparent approach for the multi-agent system is highly desirable. The visual specification of the network analytic workflows can be considered as a visual program for the control component in a blackboard-type architecture [4].

Taking all these aspects into account an infrastructure serving all these needs has to facilitate a language-transparent multi-agent system following a blackboard architecture. Thus our preferred choice for development was a middleware for distributed programming called SQLSpaces [11], an open source implementation of the TupleSpace approach [12] which focusses especially on ease of development using the server as communication platform for distributed multi-agent systems and on the support of language-heterogeneous clients. Using an SQLSpaces server allows the communication between the user interface and the computational backend as well as the communication between the individual agents of the computational backend in a loosely-coupled and extensible way.

Each single analysis module of the user interface is represented by one component in the computational backend. To initiate the execution of an analysis workflow the frontend writes so called *command tuples* into the SQLSpaces server. These command tuples include identifiers for incoming and/or outgoing pipes depending on the type of module and parameters needed for the analysis step to be executed. An example for such a parameter is a threshold value for a filtering module or a filename for a component which acts as a source feeding the content of a file into the process. Each analysis module may have one or more connectors for incoming and outgoing pipes. After receiving a command tuple, the components start reading data from the incoming pipes, processing the received data consistent with the set parameters, and write the results of the processing into the outgoing pipes. Reading incoming data or writing data is omitted for data sources or data sinks respectively. The mapping between modules in the user interface and the processing modules is done by unique identifiers. These ids are unique for each type of processing module and each module only reacts on messages which contain the information that they address that specific module.

The data exchange between two analysis modules during the execution of a workflow is also based on a simple tuple protocol. The *data tuples* exchanged between analysis components contain a unique identifier for the pipe in which they belong and the data to be exchanged.

Additionally the command tuples contain status information which is used to synchronize the agents where necessary. The possible states of the agents are *waiting*, *processing*, *finished*, and *stopped with error*. The identifiers of the pipes are connected to identifiers of the individual analysis steps. Therefore an agent can for example check if the agent(s) producing the input for it stopped with errors, shut itself down as no input is to be expected, and propagate that information further down the processing chain.

In the current implementation of the workbench the analysis modules are generally split into two tiers. One is a management component permanently connected to the communication server, the other is the actual processing agent. The management components listen for incoming command tuples signaling the request for the execution of an analysis process. As soon as such a signal is received, an instance of the actual processing agent is started, configured according to the information found in the command tuple. After an agent has finished the analysis process it was created for, it shuts down. Thus only the managers representing the available analysis

modules are running permanently, whereas the actual analysis agents are only running on demand. However this split is not mandatory. The only obligatory features of an analysis component are the connection to the SQLSpaces server as data exchange and message passing platform, the compliance with the tuple based communication protocols for receiving commands and for exchanging data, and the propagation of a filter description (see below). The processing agents may also access external services. A typical example is the access of external repositories by components functioning as data sources for workflows.

The main user interface of the Analytics Workbench is depicted in Fig. 1. It is based on a modified version of the open-source wiring editor, which is part of version 0.5.0 of the WireIt JavaScript library [13]. It consists of a list of available analysis modules on the left hand side, a menu bar on top, an information area on the right hand side, and the workspace for constructing the analysis workflow in the center. The modules are organized in several categories like *Input*, *Analysis*, or *Graph Visualizations*. The information area on the right hand side contains an overview panel for the workspace, which is by default minimized and can be expanded on demand. It is also used for showing, if the user has active processes running in the system, and to display results and error information in a history of previous workflow runs from the current web session. Results from previous sessions can be loaded using the menu bar, which also allows cleaning the workspace, loading and saving workflows, and most importantly executing the workflow which is currently in the workspace. The selection of filters visible in the user interface is created dynamically whenever a user loads the interface. The analysis components register a service description of their respective user interface module at the SQLSpaces server and deregister it when they shut down. Thus the user interface always reflects the currently available analysis modules.

As depicted in the architecture overview in Fig. 2, the communication between the user interface in the browser and the SQLSpaces and thus the computational backend

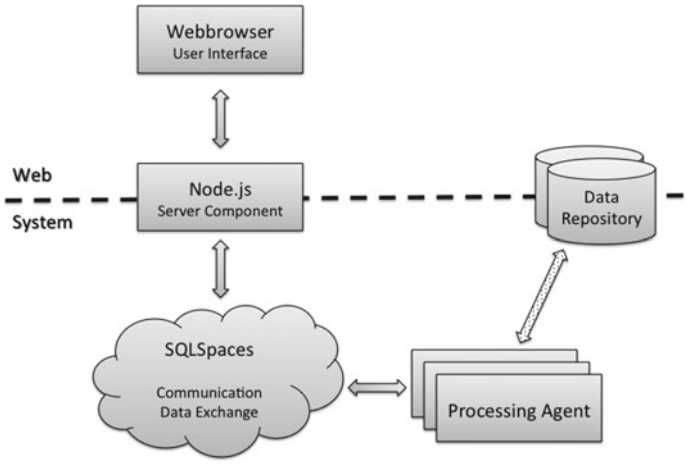


Fig. 2 Architecture overview of the Analytics Workbench

is mediated by a server component. This server component is based on Node.js [14]. It is used on the one hand for delivering the user interface to the browser of the user and on the other hand for introducing an access control layer between the web and the computational backend of the workbench. Although the SQLSpaces do offer a direct JavaScript connector it is not used here as it would offer direct access to the communication server which could not be controlled further. The server component for example allows filtering the available module descriptions before sending them to the user, so that a restriction to a certain selection of modules for a special user group would be possible. Also the execution of workflows can be controlled in such a way, that only a user who is allowed to execute all the components contained in his workflow may actually execute it. This prevents abuse of the workbench.

The use of Node.js as server component for the user interface also allows sending messages from the server to the web browser. This is used for notifying the user about the current state of workflow executions, about newly created results, and, if necessary, about errors. If the user has a workflow in the workspace, which is currently executed, the modules in the user interface are colored according to the current state of the corresponding agent instance like you can see in Fig. 1. A green module signals that the corresponding agent has finished processing, a yellow module is currently processing, and a blue module represents an agent in waiting state. A module failing and notifying about errors would be colored red.

3.1 Integration of Analysis Components

Concerning the integration of additional analysis features, there are currently two main approaches possible. One is the development of completely new components. As stated above, such components only have to be able to connect to the SQLSpaces as information exchange platform and to comply with the tuple based communication protocols. Even the data format is not predetermined although additional format conversions may be necessary for connection to existing components. The data exchange between the currently available analysis components mainly uses one of the two formats developed specifically for the use in the workbench, a graph format and a data table format. These formats are based on the JavaScript Object Notation (JSON, [15]) and have been developed with the goals to be very flexible and expressive, to allow connection between several files and the file types through shared metadata, and for ease of use in JavaScript based visualizations. In addition to that, currently the edgelist format of the network analysis tool Pajek, the Graph Modeling Language (GML), and the matrix format of the UCINET network analysis tool are supported as graph formats by conversion to and from the JSON based graph format. In terms of data table formats, converters for a variant of the Comma Separated Values (CSV) format do exist. Programming languages that can be used with a direct connection to the SQLSpaces include Java and Python [11]. If a developer wants to follow this way for extending the workbench, he mainly needs knowledge of the programming language in which the new component should be realized and of course of the actual

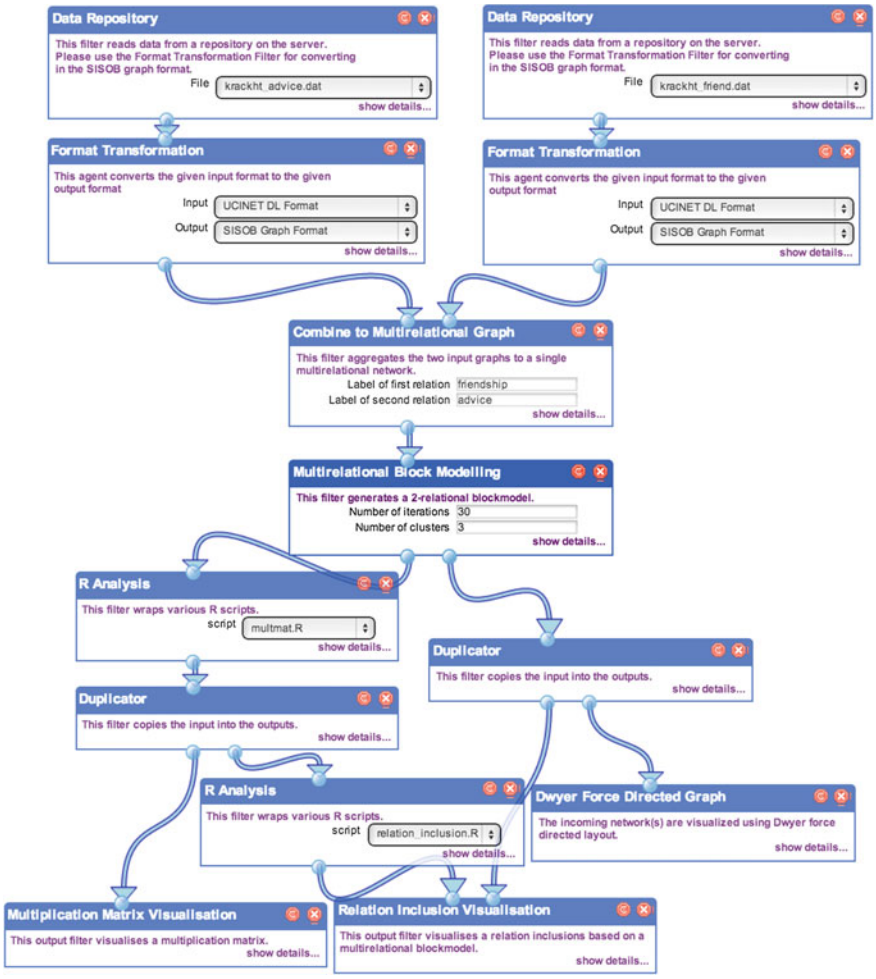


Fig. 3 Network attributed with block information (same block membership is represented via same coloring of the node) in the Analytics Workbench

analysis feature to be added. The interaction with the SQLSpaces server follows a very simple protocol and and also the interface of the SQLSpaces clients is quite simple. Thus no specific knowledge about details of programming for multi-agent systems is necessary. This is especially true for Java, where an agent framework that abstracts most of the SQLSpaces interaction is available for creating workbench agents.

The second possible option for the integration of new analysis features is the integration of scripts based on the R language [16]. This language has been developed for statistical computing and by now also offers a wide variety of matrix and graph computation functionalities, including the availability of the igraph library [17]. One

of the currently available components for the Analytics Workbench is an *R-Analysis* component. It is a wrapper for the execution of R scripts which also transfers input and output of the script from and to the SQLSpaces. The scripts to be included have to comply with a certain way of sharing input and output between wrapper and R executor through the file system. If they do this and do not need configuration parameters, the scripts can simply be put into a repository for R scripts, will be detected by the R-Analysis component and can then be used directly. In case additional input parameters are necessary, the R wrapper has to be extended for that specific script in order to provide the appropriate user interface and to transfer the parameters from the user interface to the script. Thus the integration of R scripts that do not need specific parameters mainly requires knowledge of the R programming language. In case parameters are necessary, additionally some knowledge in Java programming is needed for creating the extended R wrapper agent.

3.2 Integration and Distribution of Visualization Features

Due to the architecture of the Analytics Workbench being based on a multi-agent architecture for processing components and a web-based user interface, visualization components have two layers: The (usually Java based) agent used for connecting to the system and the web technology based user interface for displaying the actual visualization to the user.

The directly connected agent is responsible for taking the data to be visualized and preparing it for the web technology based user interface part. It prepares the data and drops it together with the files needed for the user interface into the result repository, from where it is delivered to the web browser of the user. The distribution of work between the agent preparing the data and the web technology based user interface is not predetermined and can be designed based on the needs of the respective visualization techniques. The possible approaches for the distribution of work can be partitioned into three different categories.

One extreme approach is a visualization component with purely client side computation. In this case the agent only drops the data to the result repository and all layout related calculation and the rendering of the visualization is done by the web technology based user interface code in the web browser of the user. This approach usually makes use of a JavaScript based rendering engine for interpreting the data. All of the currently available statistical visualizations fall in this category, together with some of the graph visualizations like the Dwyer Force Directed Graph technique, for which you can see an example in Fig. 3.

The other extreme would be visualization techniques that do the entire layout and rendering server side in the agent and only send non-interactive pictures to the web browser of the user. Examples for this approach are the R based visualization techniques described in this paper, like the multiplication matrix visualization (see Fig. 4) or the integrated visualization of positions, roles, and groups (see Fig. 5). In these examples, the resulting visualizations are offered as PDF files to the user.

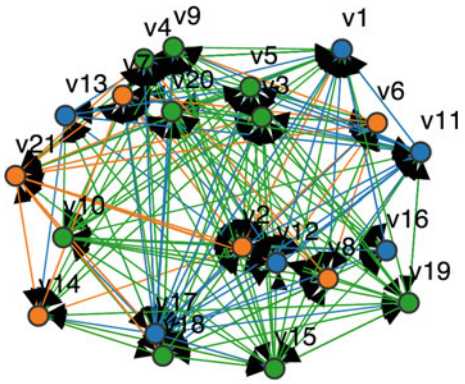


Fig. 4 Role analysis of Krackhardt's High-Tech Managers—multiplication table

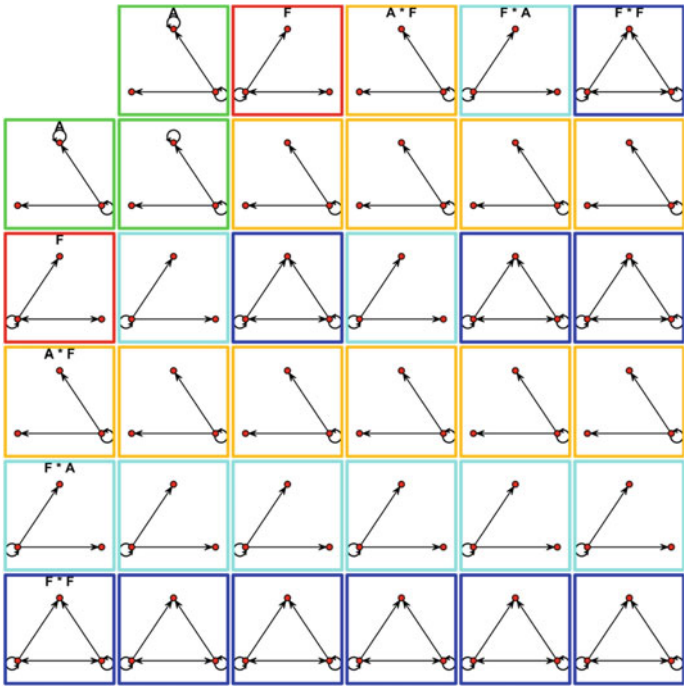


Fig. 5 Integrated visualization of positions, roles, and groups for relations F, $F * F$, $F * A$

Another similar option would be generating static images in the processing agent and offering them to the user in a web gallery.

The third category contains all those approaches where some parts of the processing are done server side and other parts are done client side. A prominent example for this category available in the Analytics Workbench is an implementation of fore-

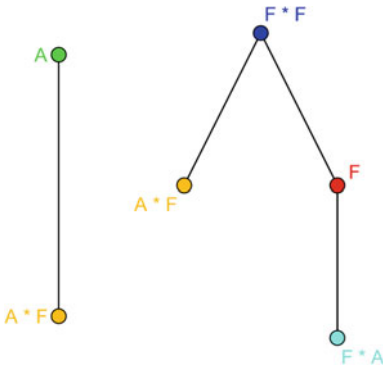


Fig. 6 Analytic workflow of multi-relational blockmodeling and role analysis represented in the Analytics Workbench

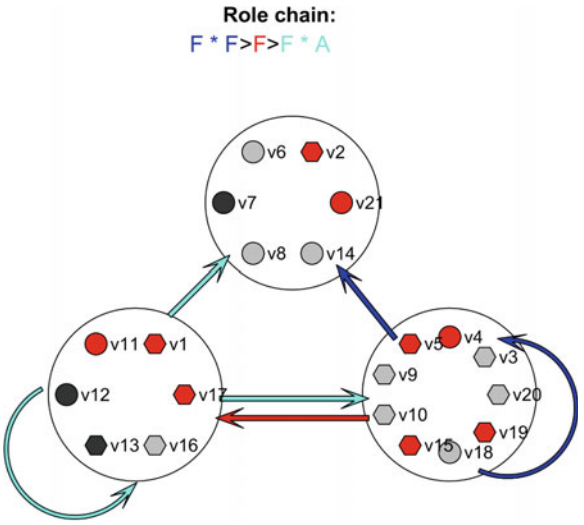


Fig. 7 Role analysis of Krackhardt's High-Tech Managers—Hasse diagram

sighted graph layout by Diehl, Görg, and Kerren presented in [18]. In this case the calculations for the foresighted layout are done server side, as they are potentially very demanding both in terms of processing power and working memory, which are restricted within the web browser. After the basic layout has been calculated, the data annotated with this information is used as input for a JavaScript based rendering engine, so that users still can interact with the final graph visualization. Thus it is possible to still use functionalities like searching for elements in the graph or highlighting of elements based on measures calculated during the analysis process.

4 Supporting Complex Social Network Analysis Processes

Network analytic processes typically consist of a series of activities that a researcher with long experience might not even think about step-by-step. This is already true for a simple workflow like the one depicted in Fig. 1. That workflow describes a four-step process. It starts with importing data into the workflow, in this case network data. The next step is transforming the input data into a format that is understood by the following analysis steps. This simple analysis process contains only one step that actually performs analysis calculations, which is the third one, the calculation of a centrality measure on the nodes. The last step of the workflow is a visualization filter, which will allow the user to have a graph visualization that is capable of showing the network and the calculated measures. Yet, some of the network analytic processes are so complex that they tend to be error-prone or awkward to replicate over and over again manually. The analytic workbench allows the re-use of analytic processes across different data sets, so that the researcher can concentrate on the activities that require the highest competences, namely interpretation and productive exploration of network data. Although the workflow described above is a quite simple one, it could already be extended to more complex workflows by adding additional analysis steps or by adding pre-processing steps like transforming author-publication networks into co-authorship networks or like filtering a network based on certain node- or edge-properties. One of the in general more complex processes to understand network structures and the relational system is the combination of blockmodeling [19] and role analysis [20]. While some of the well-known SNA tools, such as Pajek and CONCOR, support the analysis of blockmodeling in different variations, the full process including role analysis is not supported by any tool we know of, especially with complex data sets that are multi-relational.

4.1 Analytics of Multi-relational Networks

Multi-relational networks [21] are of high relevance as a formal model for social networks that represent different types of relations. Almost all social software systems and many other contexts, where network analytic approaches are applied, offer different kinds of interaction that is naturally represented as multiple relations on the same set of actors. For example facebook data can be considered using one relation for the friendship of two actors with each other and another relation for two actors being members of a social group. Reducing this to a one-relational network by aggregation usually simplifies a network too much for an appropriate interpretation. Thus, methods taking into account multiple relations at the same time would be called for, yet are much less supported in SNA tools than for the simpler network types. Blockmodeling is a technique that helps to identify general stuctures of a network by identification of actors with a similar position, i.e. similar ties to other actors. This is especially valuable for very large networks, where the sheer

number of actors denies a proper interpretation of the network structure. Role analysis allows the identification of patterns in the relational system of multi-relational data; e.g. in a political network with the relations "is friend of" (F) and "is enemy of" (E) role analysis can identify if the traditional proverb "the enemy of my enemy is my friend" holds for a specific network. If yes, the formal relation $F = E * E$ or $F \supset E * E$ would hold. Multi-relational analysis is currently not advanced with respect to algorithmic methods nor well supported in SNA tools.

The multi-relational analysis process we will present in the following subsections is following the analysis of Wasserman and Faust [21] that they conducted with an indirect blockmodeling approach on two relations and role analysis based on the resulting image matrices. To evaluate our own direct blockmodeling approach against the original analysis and apply our visualisation proposal we conducted in [22] a replication study and a comparison with our own methods. This helped us to identify a different blockmodel and substantial interpretation that is also supported visually by our diagrammatic presentation.

Yet, the authors had to build their own libraries to conduct their analysis and used several manual steps in this process that are both error-prone and time-consuming. Thus, the designers of the Analytics Workbench and the multi-relational SNA approach joined forces to allow the re-use and automated application of this complex network analytic process inside the Analytics Workbench. In the following sections we will present the formal description of the analytic process in the notation used by the Analytics Workbench and the implications on implementing existing code for sub-steps of this process as re-usable components of the Analytics Workbench.

4.2 The Formalised Model for Multi-relational Blockmodeling and Role Analysis

In order to allow the re-use of components already existing in the workbench and a flexible use of new components needed for multi-relational blockmodeling into the workbench in different contexts, the original integrated analysis script for block-modeling was split into several components. Figure 6 depicts the resulting workflow. It starts with separately loading two different graphs from a repository, converting them into the appropriate format for further processing in the Analytics Workbench, and integrating them into one multi-relational graph. For integrating the graphs into one multi-relational graph, the two graphs need to have the same nodeset. The component also allows combining two multi-relational graphs with each other or a multi-relational graph with a single-relational graph. Thus it is possible to collect an arbitrary number of relations by cascading the component.

The next step in the analysis process takes the multi-relational graph as input and performs the actual blockmodeling algorithm on it. There are two main results of that process. One are the resulting image matrices represented as data tables, the other is the original multi-relational graph annotated with the position for each

actor as determined by the blockmodel. This annotated graph is fed through a duplicator, which just copies the data into two pipes, and is then visualized using a force directed graph visualization already available in the workbench. Figure 3 shows the result of that visualization process. At this point we make use of combining newly integrated analysis components with existing components of the workbench to visualize a result of the analysis process for which no visualization was foreseen in the original implementation of the multi-relational blockmodeling approach.

The image matrix resulting from the blockmodeling process is piped into an R-Analysis filter which executes a script calculating the equivalence classes constructed from successive composition of all relations derived from the image matrices resulting in a multiplication matrix. This matrix is also duplicated with one of the duplication output pipes leading into a module for visualization of the multiplication table. The other analysis done on the multiplication matrix is calculation of the dependencies between the different equivalence classes. The result of this is combined together with the annotated version of the original multi-relational graph in a third visualization component, which produces integrated visualizations of positions, roles, and groups as in Fig. 5.

4.3 Implementation Issues for Extension of the Workbench for This Process

As explained in Sect. 3 the Analytics Workbench can be customized either by writing new agents or, which is more simple, by putting custom R scripts into a R script repository. As the original implementation of the multi-relational blockmodeling approach is based on R, using the R-Analysis filter is the approach used here. In order to be able to use single steps of the analysis process separately, the blockmodeling script was split into several parts as described above. However some of these parts need additional parameters. As the plain R-Analysis filter does not support additional parameters for the R scripts, simply putting the resulting small scripts into the filter's script repository was not a feasible solution for all analysis steps. For those scripts which need additional parameters, new filters have to be integrated in the analytics workbench. It is, however, still possible to use custom R scripts for this task to keep the implementation effort small. Since the basic R-Analysis component itself is just a Java based wrapper for R scripts, new R based agents can inherit the core functionality from it. This allows the construction of a parameterizable R-Analysis agent for a special purpose. During the developement of our workbench this procedure has become common practice, e.g. for the k-core extractor filter.

The picture of the workflow in Fig. 6 shows clearly which implementation strategy is needed. The first two new filters, namely *Combine to Multirelational Graph* and *Multirelational Block Modeling*, both require the specification of parameters. Thus, two new components based on the R-Analysis component can be used. These components have a customized user interface allowing the necessary parameterization

for the respective script. The resulting image matrices of the Multirelational Block Modeling filter can be represented as a set of data tables and hence suit the input requirements of the R-wrapper. To build the multiplication matrix based on these image matrices no additional parameters are needed and consequently the corresponding part of the original R script can be directly fed into the script repository without further adjustments. The same is true for the relation inclusion script.

The custom visualization components used for displaying results of the multi-relational blockmodeling process however again need specialized agents. Again these filters are based on R scripts, but in this case the reason is not the need of additional parameters. As these filters create visualizations, their output is not fed back into the workflow but offered to the user as viewable results.

4.4 Summary of the Complex Network Analysis on "Krackhardt's High-Tech Managers"

This example is one of the well known multi-relational datasets used to compare and validate algorithms and methods of social network analysis. It has been well researched and published and was chosen originally [23] to be contrasted with the elaborate analysis in [21]. Since our focus in this paper is to demonstrate the analytic flow and not the comparison with others' analyses, we will not go deeply into details on the substantial level and the comparison performed in [22].

The network data was collected in a high-tech company with 21 managers as actors of the network. Collected data represents three directed and binary relations: REPORTS_TO, ADVICE, and FRIENDSHIP. Additionally, several actor attributes have been gathered, namely the age of the manager, duration of employment in the company, hierarchy level, and department. Because Wasserman and Faust presented the methods of position and role analysis based on the ADVICE (A) and FRIENDSHIP (F) relation, we also focused the analysis on these two relations. Yet, both the original design in [23] and the transfer to the Analytics Workbench are not constrained just to two relations, the third relation could be taken into account, yet would make the analysis even harder to explain in condensed space. While in the original approach the analysis including the third relation would have to be performed again in the step-wise manually triggered fashion, this could be done automatically in the Analytics Workbench by loading the third relation into one of the data sources, i.e. a direct re-use of the workflow with a differently parameterized dataset.

As described in [22] we applied the direct blockmodeling method with different relation-dependent densities to generate image matrices. This has been achieved by the analysis workflow shown in the top half of Fig. 6 with its two data sources (the two relations *Advice* and *Friendship*), transformation into the workbench's internal data format and the new component for multi-relational blockmodeling integrated as an R script.

The resulting blocks of the best fitting blockmodel with three positions, that can also be considered a specific clustering of actors in the same position/block, can then be visualized with existing visualizations such as the force directed graph view shown in Fig. 3.

Applying our role analytic method [22] to the resulting image matrices via the matrix multiplication method we obtain five different equivalence classes shown in the R visualization of Fig. 4. These classes are considered as input for the next analysis component, the relation inclusion. The inclusion dependencies detected as global roles are shown in Fig. 7 that also has been implemented as an R visualization.

The visual integration of positional analysis, global roles, and group information is conducted as PRoG diagrams (Positions, Roles, and Groups) described in [22]. Figure 5 shows the PRoG diagram for the positional membership of each actor and the role chain consisting of the inclusions $F * F \supset F$, $F * F \supset F * A$, $F \supset F * A$.

5 Benefits of the Workbench and User Evaluations

As mentioned above the workflows produced with the Analytics Workbench can be **directly re-used** as is for similar datasets. In the example case of the Blockmodeling and Role Analysis other multi-relational datasets can be analyzed immediately just by using the data sources with different relation data. This allows comparative studies of network structures using exactly the same analytic methodology.

On the other hand, a researcher might be interested to apply different analytic steps, exchange algorithms (i.e. for clustering, group detection) or explore networks with different visualizations. All this is possible by **adapting** parts of the analytic workflow by inserting or exchanging analytical flow components or adding visualizations to compare the insights gained through the graphical inspection.

The Analytics Workbench has undergone a user evaluation in a one-day evaluation workshop conducted in context of the SISOB project [24, 25]. The five invited participants of the workshop are all experts in network analysis and/or scientometric analysis and thus represent the range of users of the workbench expected in the SISOB context. Main goals of that workshop were testing the usability of the workbench, and gathering information about the perceived usefulness of the workflow representation. For that purpose, the main part of the workshop was organized in two experimentation phases. For these experimentation phases the participants were split in three groups (either pairs or individuals). Each group conducted each phase accompanied by a demonstrator who also acted as interviewer and note-taker for a semi-structured interview. Each group was equipped with a computer that was used to access the workbench.

The workshop started with a brief introduction of the workbench to the plenum of all participants. Afterwards, the first experimental phase was conducted as an exploration of the Analytics Workbench guided by the demonstrator. In itself it was again split in two phases. The first part was an introduction to the workbench. Each demonstrator explained the workbench to the participants by creating, executing and

modifying workflows. In the second part analysis tasks were given to the experts and should be performed by them using the workbench. This first phase was concluded by a semi-structured interview aiming at finding usability issues. It contained both questions specific to the workflows and filters related to the given task as well as more open questions on general comments about the workbench.

In the second phase the participants were asked to describe typical analysis processes in their daily work and to try to map them to the workbench. For this mapping they were first asked if and how the processes could be mapped to the workbench version and filter selection that was available during the workshop. If they could not map an analysis process to the workbench directly, they were asked to explain, which functionalities were missing and how they would like them to be implemented as filters. Additionally they were asked if and how they could imagine using the workbench in their daily work, and which benefits or problems they would expect.

The results of the first phase showed several non-severe usability issues which have been the basis for improvements of the workbench in the mean time since the workshop has been conducted. One example for such an issue was the ordering of the filters in the filter menu, which they would have liked to be ordered in the typical sequence of a workflow. One result of the second phase was apart from several suggestions for additional filters to be included also one complete workflow mapped onto the workbench as it was used during the workshop. In terms of experienced and expected benefits of using the workbench, the participants confirmed those described above. Some of the experts also expected improved communication as additional benefit. They regarded the visual representation of the analysis workflows as very suitable for explaining the workflows themselves, how specific analysis results have been produced, and how these results relate to the original data they are based upon.

6 Summary and Outlook

In this paper we presented the concept of a web-based analytics workbench to support researchers of social networks in their analytic processes. Important features of this workbench are on the one hand making analysis processes explicit by using a visual representation of the workflow and on the other hand the flexible technical framework behind it which allows extending the workbench with new analysis techniques easily. We have used a recently developed blockmodeling approach for multi-relational networks as a showcase for both of these features. We have explained how the existing implementation of the analysis approach can be integrated into the workbench and we have used the visual representation of the example workflow as a means to help explaining the overall analysis process and to describe its individual steps. We also explained the results of the presented analysis workflow.

Future work on the Analytics Workbench will include using the workbench in teaching of network analysis and an evaluation if the visual representation of workflows helps the learners in understanding the analysis processes. Another direction

of work on the Analytics Workbench will be improving the extensibility for non computer experts. For that purpose the R-Analysis component described in this paper will be combined with a user-accessible repository and with support for authoring R scripts, that are compliant with the interfaces of the R-Analysis component. Thus also analysis experts with experience in the R language will have the opportunity to extend the Analytics Workbench for their specific needs. We invite both researchers interested in using the workbench and developers interested in contributing to visit http://workbench.collide.info, where they can find a demo installation to try the system and also further information.

Acknowledgments The development of the Analytics Workbench presented in this paper was partially conducted in the context of SISOB [24], which is funded by the European Community under the Science in Society (SIS) theme of the 7th Framework Programme for R&D (Grant agreement 266588). This document does not represent the opinion of the European Community, and the European Community is not responsible for any use that might be made of its content. We want to thank Alona Schmidt for allowing us to use her original implementation of the multi-relational blockmodeling approach.

References

1. http://pajek.imfm.si/doku.php?id=pajek. Accessed 15 April 2013
2. NWB Team (2006) Network workbench tool. Indiana University, Northeastern University, and University of Michigan. http://nwb.slis.indiana.edu/
3. Mell P, Grance T (2011) The Nist definition of cloud computing: recommendations of the national institute of standards and technology (Special publication 800-145 (20.10.2011))
4. Buschmann F, Meunier R, Rohnert H, Sommerlad P, Stal M (1996) Pattern-oriented software architecture, volume 1: a system of patterns. Wiley, New York
5. http://www.omg.org/spec/UML/. Accessed 15 April 2013
6. Petri CA (1962) Kommunikation mit Automaten. PhD thesis, Fakultät für Mathematik und Physik der Technischen Hochschule Darmstadt, Institut für Angewandte Mathematik der Universität Bonn, Wegelerst. 10
7. http://rapid-i.com/content/view/181/190/lang,en/. Accessed 14 April 2013
8. http://pipes.yahoo.com/pipes/. Accessed 14 April 2013
9. Bigaret S, Meyer P (2010) Diviz: an MCDA workflow design, execution and sharing tool. Newsl EURO Work Gr Multicriteria Aid Decis 3(21):10–13
10. Herr B, Huang W, Penumarthy S, Börner K (2006) Designing highly flexible and usable cyberinfrastructures for convergence. Des Highly Flex Usable Cyberinfrastruct Converg 1093(1):161–179
11. Weinbrenner S (2012) SQLSpaces—a platform for flexible language-heterogeneous multi-agent systems. PhD thesis, Universität Duisburg-Essen
12. Gelernter D (1985) Generative communication in Linda. ACM Trans Program Lang Syst 7(1):80–112
13. http://dev.lshift.net/james/wireit/wireit/index.html. Accessed 14 April 2013
14. http://nodejs.org/. Accessed 14 April 2013
15. http://tools.ietf.org/html/rfc4627. Accessed 14 April 2013
16. http://www.r-project.org/. Accessed 14 April 2013
17. http://igraph.sourceforge.net/. Accessed 14 April 2013
18. Diehl S, Görg C, Kerren A (2000) Foresighted graphlayout. Technical report, Universitäts- und Landesbibliothek, Postfach 151141, 66041 Saarbrücken

19. Doreian P, Batagelj V, Ferligoj A (2005) Generalized blockmodeling. Cambridge University Press, Cambridge
20. Winship C, Mandel M (1983–1984) Roles and positions: a critique and extension of the block-modeling approach. Soc Methodol 14:314–344
21. Wasserman S, Faust K (1994) Social network analysis: methods and applications. Cambridge University Press, Cambridge
22. Harrer A, Schmidt A (2013) Blockmodeling and role analysis in multi-relational networks. Soc Netw Anal Min 3(3):701–719
23. Harrer A, Schmidt A (2012) An approach for the blockmodeling in multi-relational networks. In: International conference on advances in social networks analysis and mining. IEEE CPS, pp 590–597
24. http://sisob.lcc.uma.es/. Accessed 6 March 2014
25. Göhnert T, Hoppe HU, Pfützenreuter N (2013) Deliverable 6.3: configuration, test of the platform and first evaluation report. Technical report, SISOB project

On the Usage of Network Visualization for Multiagent System Verification

Fatemeh Hendijani Fard and Behrouz H. Far

Abstract Multiagent Systems (MAS) consists of many software agents that interact to each other to perform their actions and achieve system goals. Due to the growing demand of Distributed Software Systems (DSS) and MAS as a branch of DSS, the verification of these systems has taken a special attention. The verification of these systems is required because MAS and DSS are large scale systems, where a failure can result in a huge amount of cost or damage. One major field is verification of MAS designs to prevent cost of fixing problems after implementation and deployment. The agents and interactions among them form a network of agents. This network can be used for verification of MAS from different perspectives and by various techniques. Visualizing agents' networks can lead to detect a special type of unexpected behavior in MAS referred to as emergent behaviors and implied scenarios. These unexpected behaviors are more probable in large scale systems because the functionality and control are distributed and there is lack of central controller in MAS and DSS. Consequently, a new scenario can be implied to the system at run time which may not be an acceptable behavior in the system. In this article, visualization techniques are applied to form three different networks extracted from the designs of MAS for this purpose. These networks are used to detect emergent behaviors and implied scenarios in the system. The methodology, system architecture, data preparation and visualization, required network definitions, and results are verified through case studies.

Keywords Network visualization technique · Multiagent systems verification · Scenario-based software engineering · Emergent behaviors · Implied scenario

F. Hendijani Fard · B.H. Far (✉)
Department of Electrical and Computer Engineering, University of Calgary, Calgary, AB, Canada
e-mail: far@ucalgary.ca

F. Hendijani Fard
e-mail: fhendija@ucalgary.ca

J. Kawash (ed.), *Online Social Media Analysis and Visualization*,
Lecture Notes in Social Networks, DOI 10.1007/978-3-319-13590-8_10

1 Introduction

In recent years, industrial applications of agent based systems have gained popularity due to certain characteristics of agents, such as autonomous, collaborative, and proactive characteristics [1]. Other characteristics, such as information hiding, ability to learn, knowledgeability, autonomy, and mobility, are among features that have made them ideal for use in Distribute Software Systems (DSS) [2]. One of the problems that arise in DSS and Multi Agent Systems (MAS), is the lack of a central controller that may cause the components/agents a new behavior to emerge at run time, which was not seen in the system specifications. This unexpected behavior, which is known as emergent behavior at the *component level* (i.e. considering behavior of agents individually) and implied scenario at the *system level* (i.e. considering the system behavior), may cause critical damage [3]. Since these are unexpected behaviors, which were not seen in the designs, they should be detected and verified before system deployment. The acceptance or denying these behaviors is what the software designer and stakeholders should decide on. However, they should be aware of such behaviors earlier to consider these behaviors in the design or change the designs to avoid them. The early detection of these behaviors reduces the cost and time for fixing them [4]. It is stated that "*Detecting the causes of faults early may reduce their resulting costs by a factor of 100 or more*" [5].

There are many approaches for the detection of emergent behaviors in the requirements and design phases of scenario based software systems. In these systems, the agents/components' behaviors and their interactions are shown through graphical elements such as Messages Sequence Charts (MSC) or UML Sequence Diagrams. In this paper, the MSCs are used as inputs of the system since they are widely used for DSS. Figure 1 is an example of presenting system scenarios in the form of MSCs. These are parts of the scenarios related to a multiagent system for managing a greenhouse. The agent A_w is the agent responsible for irrigation of plants. This agent interacts with other agents, such as A_t and A_m (agents responsible for heat balancing and providing minerals) to perform effectively. Suppose that the only acceptable interactions among agents of this system are shown in Fig. 1.

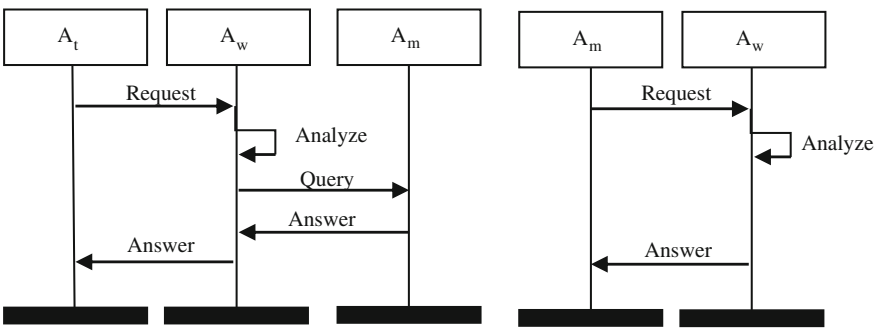


Fig. 1 Two scenarios showing part of agents' interactions for Greenhouse MAS

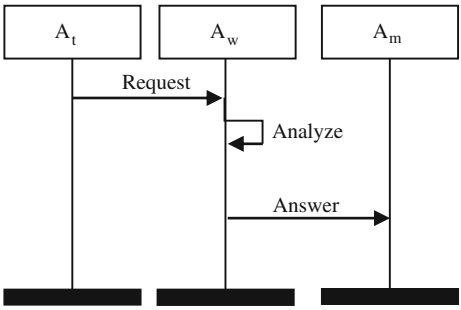


Fig. 2 An emergent behavior in Greenhouse MAS

The acceptable actions of agent A_w are shown in the MSCs of Fig. 1; however, this agent may behave differently as shown in Fig. 2. Therefore, if a new scenario, such as Fig. 2 happens, it can lead to a problem in the system.

Figure 2 is an example of an emergent behavior. From the point of view of agent A_w, this agent has a choice to follow, when it receives the "Analyze" message and reaches its third state. Therefore, it neither follows the first MSC nor the second one. This behavior is called an emergent behavior of A_w. More explanations as to the reason and the effects of this example are presented in next sections. It is worth mentioning that the decision of accepting or declining implied scenarios and emergent behaviors depends on the stakeholders of the system, but they should be detected and defined earlier to redesign the system to handle these situations.

Most research in the detection of emergent behaviors and implied scenarios use model checking approaches that try to verify whether the model explaining the behavior of components/agents satisfies the requirements and designs. These approaches use various behavioral modeling techniques and finite state machines to be verified against the extracted languages [6–8]. One of the issues with model checking approaches other than the problem of state space explosion (when they face large scale systems) [9–16], is that most of them cannot find the exact point or cause of emergent behavior and just identify the existence of an emergent behavior in the system [17]. Therefore, this is a flaw that these approaches do not suggest solutions to the designer and do not fix the problem either by the designer or automatically.

In this paper, visualization of the agents' networks is taken into account as a major tool for emergent behavior and implied scenario detection. Three different networks are extracted from MSCs and visualized to detect the points in which an unexpected behavior can happen. It can both demonstrate where and why the unexpected behavior can occur and provide the reasons for and the effects of emergent behaviors. One of the networks is associated with events on any individual agent for component level analysis and the other two networks illustrate the agents' interactions with each other and how they see the sequence of actions that should be performed at the system level. These networks are used to show violations to the designers and stakeholders and to verify the behavior of agents. Although, in this paper, the complexity of existing approaches and the present work is not the main issue, in our technique, model checking approaches are avoided since they require behavioral modeling or synthesis

phase, which is considered to be even more complex than model checking [18]. To avoid synthesis, we have devised a new technique for emergent behavior and implied scenario investigation [19, 20] using interaction matrices. The scalability issues of this technique is explained more in [21]. In this paper, the interaction matrices are used to extract and demonstrate the three mentioned networks: agent's states network, agents' interaction network, and graph of sequence of MSCs for each agent. Based on our knowledge, this visualization for MAS verification against emergent behavior and implied scenarios is among contributions of this research.

It is worth mentioning that these networks are semantically and visually different from the state machines in Finite Automata and formal methods. This is more explained in the Discussion part of this paper. The related works and basic information about scenario based systems and MSCs, emergent behavior and its detection approaches, and applications of social network analysis on MAS verification are presented in next section. It is followed by the methodology which explains system architecture, general steps of methodology, component and system level analysis, data preparation, data visualization, verifying the networks, and how to generate reports in Sect. 3. Two examples are presented to illustrate the methodology and results verification. The first example is a MAS greenhouse, which is used to explain component level analysis. The second example is from Uchitel's et al. [22] and is used to explain the system level analysis. In the rest of this paper the words *component* and *agent* may be used interchangeably and by both words we mean the word *software agent*.

2 Related Works

2.1 Scenario Based Systems and Message Sequence Charts

In software engineering, a practical approach is describing software requirements using scenarios. Scenarios are stories about people and interactions, including agents and actors and sequences of actions and event. Scenarios can describe different levels and details with multiple perspectives. For different purposes, scenarios can provide abstraction for the designer and problem solving [23]. The visual forms of scenario based specifications like UML Sequence Diagrams (SD) and Message Sequence Chart (MSC) can show the software behavior. Software system components (processes) or inter-object and inter-processes are shown using vertical lines with their interaction messages and send/receive events respectively in MSC or SD [24, 25]. These specifications can be used for model checking, formal verification, or monitoring for emergent behavior detection [26].

The MSCs are visual and formal description techniques for software requirements and are widely used for MAS and DSS [27–30]. DSS has gained a broad attention in recent years for developing various systems and a main process in DSS is how

to show the system components and their cooperation [29]. This process in many works is done using MSCs. Development of MSC, its definitions and semantics has been standardized and published by Telecommunication Standardization Sector of International Telecommunication Union (ITU) [31]. MSC describes the communication behavior of system components and their environment in the form of graphical representation and message interchanging.

In an MSC each vertical line shows a component and the messages between them are shown with arrows. The name of the component is written above its line. Each message is shown with the label above it. The related scenario shows the MSC name. Figure 1 shows an example of a MSC.

A formal description of MSC from [32] is defined as a set (E, L, C, μ, $<$, M) where:

- E is the set of events consists of send and receive ones.
- L is a finite set of labels
- C is the set of components
- μ is the mapping of events to labels and components
- $<$ is the set of total orders on the events and μ
- M maps the send events to receive ones

In an MSC, the messages on the life line of each component are referred to as events on that line. These events can change the state of a component. The network of these states for each component is referred to as agent's state in the rest of this paper. Going from one state to another state or changing the state for an agent occurs by receiving or sending a message to that agent. The message (event) that changes the state of an agent is referred to as state transition in next sections.

The graphical means for showing how a set of MSCs can be combined are shown either with Message Sequence Graph (MSG) or high level message sequence charts (hMSC). These two are structured models of MSC. MSG has operations such as choice and repetition. MSG has MSCs as its nodes and the edges represent the concatenation of MSCs. A choice or branching in MSG can be shown with a hexagonal. An hMSC is a graph like formation and each node can be either an MSC or an MSG. In hMSC, the start node is shown with an upside down triangle and the end MSCs is shown with a triangle. In many researches MSG and hMSC are considered almost the same. MSC, hMSC, and MSG can be considered as the early models of the system [33].

2.2 Emergent Behavior

Emergent behavior is defined as the unexpected behaviors of software components in the execution time, while it was not seen in their requirement and design specification [4]. These behaviors may cause irreparable damages to the system and therefore detecting them in early phases of software lifecycle is valuable [6, 34]. Emergent behaviors are also referred to as implied scenarios i.e. when integrating all of the

scenarios of the system (e.g. in the form of state machine), they may imply a new scenario or unexpected behavior. This happens because the model is not the precise equivalent of the requirements specification [17, 35, 36].

Emerging new behaviors in DSS is more probable because there is lack of central control in these systems. This reason causes the components to have a local view of the system. Consequently, they may start with one scenario of the system and continue in another scenario in a shared state [37–39]. This state in some researches is referred to as identical states [4].

The problem of detecting emergent behavior can be classified in two categories which have different names based on the phase of software lifecycle and the scope they are used. Component level emergent behavior detection investigates the unexpected behaviors in component level i.e. considering the individual component and not its interaction with other components of the software. This refers to component fault in some researches where the behavior of the implemented component does not satisfy its specified behavior [5]. The second class is the detection of emergent behavior in system level i.e. investigating the components and their interactions as a system. Because even if the components have no emergent behavior, they can behave differently when they are integrated and have interaction with other components of the system. This is referred to as emerging an implied scenario in the system. The reason of having implied scenarios is explained through an example in next sections.

2.3 Emergent Behavior Detection Approaches

The existing approaches consider detection of emergent behavior, implied scenarios, or both. One approach for model checking in the requirement is Alur et al. methodology [33]. His works define a detailed explanation of model checking of Message Sequence Chart (MSC), Message Sequence Graph (MSG), and high-level Message Sequence Chart (hMSC) [6]. All of the linearization of this model (MSC and hMSC) is then checked against an automaton which is defined by the message alphabet $\sum$, words, languages, states, and transitions which show the behavior of each process/component. If the intersection of the automaton (showing unwanted behavior of the system) and the linearization is empty, it shows that the defined model with MSC satisfies the requirements. He also defines the realizability of MSC and MSG which means the implementation should generate the exact behaviors specified in the graph. The safe realizability has polynomial-time solution for MSC [40]. One problem with model checking with Finite State Machine (FSM) is the required processes to prepare the model: Flattening, remove cycles, etc. [41].

Whittle and Schumann et al. propose a methodology which uses Unified Modeling Language (UML) notations and investigates the conflicts in translation between UML notations. They define a methodology with algorithms in different steps supported by a tool to synthesize statecharts from Sequence Diagrams (SD), class diagrams, and Object Constraint Language (OCL) specifications. Their main focus is on using their methodology for Agent Based Systems (ABS). In this translation, they detect

conflicts and indentical states in merging the scenarios to have justified merging of scenarios [30, 42]. Their most work is on the design part or synthesizing scenarios to state machines and code [43, 44].

Ben-Abdallah explains the non-local branching choice which causes the emergent behavior in system. This problem may arise from the abstraction view of MSC and not having semantics or specific interpretation of the implemented process when MSCs face a branching choice in hMSC [45, 46].

Muccini inspects the effects of non-local branching choice on emerging implied scenarios and explains the reason of an implied scenario generation. When the state machines are synthesized from scenarios (which explain a model and specification of the system), a set of behaviors are presented by state machines which was not in the scenarios [35]. In the non-local choices, different processes send first events in different branches. This is detected by "augmented behavior" of processes in the non-local choice in their algorithm. When processes share the same augmented behavior in the non-local choice, their interaction generates an implied scenario.

Song et al. methodology is quite different with the other approaches. They explain the detection of implied scenarios when using UML as the specification language. They generate two graphs named specification and implementation graphs. By matching these two graphs, the implied scenarios and the exact cause of its emergence in the specification are identified. Their methodology considers both synchronous and asynchronous communications [17, 47].

Uchitel et al. provide a methodology and tool for detection of implied scenarios. Their algorithm builds Labeled Transition Systems (LTS) for behavioral modeling of MSC and hMSC (as semantics of MSC) with synchronous communication. They have implemented their work in a tool named Labelled Transition System Analyzer (LTSA) [8, 22, 37, 48, 49]. The work is later extended by Letier et al. to detect input-output implied scenarios [50]. They also refer to the rejected implied scenarios by the stakeholders as negative scenarios and define behavioral constraints with LTS for eliciting them from implied scenarios. Kramer et al. use Timed automata and the behavior model specified by LTS to animate the behavior models in LTSA. The Timed automata adds local clocks to LTS [51].

Kumar et al. discuss the problems with main researches in this field. He discusses that many researches are not implementable and amendable, or do not show correctness. They use MSG and FSM and define a reduced transition system to detect implied scenarios and model checking. They develop a complete method for this problem claiming the correctness and ability of implementation of their work [7]. Correspondingly, they discuss that "without message labels, if we observe one process at a time, checking for implied scenarios is decidable" [39]. There are other works presented in [52, 53] regarding the theories of message passing automata and regular MSC.

This paper uses a different approach for verifying MSC specifications using interaction matrices. Several pieces of information can be extracted from interaction matrices: number of messages (events) for each component/agent in each MSC, type of messages (send or receive), the order of messages in the MSC, the interacting agents, the senders and receivers of messages, and the states of each agent.

Using these matrices helps to visualize specific networks required for detection of emergent behaviors and implied scenarios. The network demonstration also helps to detect where the emergent behavior has happened (in terms of the events of one agent in the system), which possible MSCs it affects, and probably the origin of such a behavior. This ability will help the designer to fix the problem rather than just notifying of the existence of emergent behavior.

2.4 Social Network Analysis in MAS Communication Verification

Social networks as a basic view are a series of individuals and their interaction such as Facebook. However, many other forms of social networks exist. Social networks can be considered as various types of interactions in different communities [54, 55]. The entities are considered as nodes with edges representing links between them which can be shown in matrices [56]. Many analyses have been proposed to social networks to specify different characters and in various applications [57, 58]. Some measurements in SNA are link or node measures like tie strength, betweenness, degree, authority, and hub measure, etc. [59] which are useful in many aspects of network analysis. One application of SNA is verification of MAS communication in a network of agents. For example, some works evaluate and verify the communication patterns of MAS. They classify communications in classes like overloader, overloaded, isolated, and regular. Then they try to find design drawbacks in terms of communication patterns [60–62]. Data mining tools are developed for this purpose. SNA techniques like clustering is used for code debugging [63]. Some other data analysis tools are available that analyze the interaction among agents and visualize the result of running simulations to be compared with the modeling of agents in MAS [64].

There is much difference between these works with what is presented in this paper. Many of the researches in this field are restricted to the verification of MAS from system logs, where the multiagent system is implemented and deployed. Some like [64] work in the design phase, but they use some kind of simulation and run the models of the system to analyze the agents' interactions. The other factor that makes a difference between our work and other researches that use SNA is analyzing the communication to verify or detect different violation factors. We mostly look for the violations in terms of new behaviors in the system (whether or not they are accepted/declined by designers and stakeholder). However, these researches do not detect possible new behaviors (referred to as emergent behavior in this article) and mostly look for communication performance, interaction verification, or issues associated with quality of service.

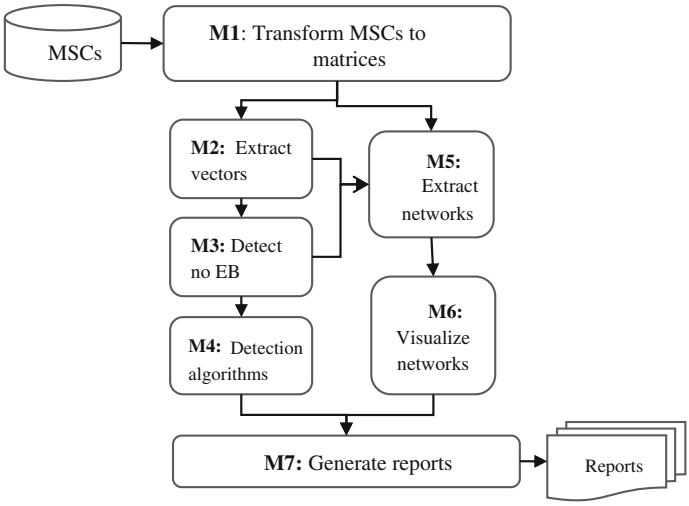


Fig. 3 Main modules of the system

3 Methodology

3.1 System Architecture

The main modules of the system are shown in Fig. 3. The MSCs are used to extract the allowable interactions among agents. The MSCs are transformed to interaction matrices in module M1. In the next step some vectors are extracted from the matrices in module M2. The required definitions can be found in next sections. In module M3 the agents with no emergent behavior are detected and removed from further analyses. The detection of agents with no emergent behavior helps in reducing the complexity of emergent behavior detection in system level. The algorithm works basically on the similarities of an agent's behavior in various MSCs to detect whether the agent is a safe agent (in terms of possibility to show an emergent behavior in component level). The behavior similarity of the agent is searched by finding all the interactions that one agent is involved in, the similarity of its states, and the messages the agent is involved in. The details of the algorithm and implementation are discussed in [21]. The next step is applying the emergent behavior algorithm in M4 for component and system level. The agents with no emergent behavior are omitted for further analysis in component level. Different algorithms are used in this module. Each algorithm detects an specific type of emergent behavior. On the right part, modules M5 and M6 are shown which extract networks and visualize data. Based on the algorithms in M4 and the networks extracted, M7 generates reports on the states, agents, and MSCs that an emergent behavior or implied scenario occurs. The report also contains information on the causes of this behavior which is also shown in the network.

The focus of this paper is mostly on M5 and M6 and the details of these two modules are presented. Therefore, details on the rest of the system are referred to other papers [19, 20].

3.2 Methodology Steps

The methodology for the detection of unexpected behavior has the following steps in general:

- Transfer the MSCs to interaction matrices.
- Transform message contents to message labels.
- Link message labels to entries of matrices.
- Extract vectors.
- Detect the agents that have no emergent behavior and omit them from processes in component level.
- Find shared states for each agent.
- Apply algorithms based on the type of emergent behavior.
- Prepare data.
- Visualize the networks.
- Apply network analysis and make them visible in the networks.
- Generate reports and compare results for the designer.

Some definitions of the matrices and the vectors, information about the networks, and how to prepare and visualize data are discussed in detail in the following for both classes of emergent behavior: component level and system level.

3.3 Component Level Analysis

In component level analysis, the detection of emergent behavior for one agent is important. In this level, an agent is considered and its states in all scenarios are investigated to detect the potential points in which the agent can be confused between two or more situations, or the states in which there is a branch with no priority to continue the next actions. This situation can be interpreted as a graph, which has an initial point, and the next node is followed when the required action for the associated edge happens in the system (events on the agent's life line). If one node has two or more different next nodes, either with same or different events on the associated edges, there is a branch in that graph. On one side, each branch should be taken into account with only one scenario of the system with required interaction among other agents. On the other side, when the software agent is traversing that graph, it has an option between the branches to choose and follow. Consequently, an unexpected situation can happen when the agent chooses another branch while the whole system expects the actions on a different branch to be performed by the agent, as it was

shown in the scenario of the system. In component level analysis, such branching choices are investigated to detect the emergent behaviors.

One possible reason of these branches in a graph is when some states of one agent are shared between two or more scenarios of the system. These states can be extracted from the interaction matrices (refer to Definition 1 for matrices). Since the state transitions are due to the messages sent/received by/to an agent, these states can form a network. The agent's states are the nodes of this network and the messages that cause a transition to another state are considered as the edges of the network. Visualizing the network of each agent's states can lead to the detection of emergent behaviors. This network, which is visualized as a directed graph, can be extracted using the interaction matrices. Each matrix is associated to one scenario, and therefore it contains the required information for a network state for each agent. Adding another scenario to the system means expanding this base network. As mentioned before, some states and their transitions may be shared among some scenarios. When expanding the network, these shared states can be visualized as the weights of a part of our network, i.e. when some states and the message labels are repeated weight of the edge between those states is increased. The more the weight, the thicker the edge is in the network. This visualization is helpful in conducting the research for detection of emergent behaviors in component level in a few aspects:

1. The shared states of each agent are shown in its states' network.
2. The branches in the graph of each agent are indicated.
3. The distance of these branches from the last state of the shared states are specified.
4. The message labels for the branching choices are illustrated.

All these choices are important when the agent behavior is verified. In case of having a branch in the network, two probable paths are:

1. Case C1: Going to two or more different states with the same message labels (edges)
2. Case C2: Going to two or more different states with different message labels (edges)

The former can be a design flaw if no condition is considered with the same messages. The latter is the case declared previously and can cause emerging a new behavior in the system. These two concerns can occur after the last state of the shared states or in case where there are no shared states between some scenarios.

It is worth mentioning that the agent should be a sender in all cases. It means the messages should be of type "sending" when the agent reaches a branching choice. These cases are demonstrated in the Fig. 4.

As it is shown in the figure, the branching occurs after state S_1. By receiving message label m_1 there are two available states that the agent can follow, S_2 and S_3. This is the path indicated in Case C1. The path in Case C2 happens when the next states can be obtained by receiving message labels m_2 and m_3 which goes to unique states S_4 and S_5 by following each of them.

Following our example of greenhouse MAS from introduction (Fig. 1), from A_w point of view, and based on its states' graph, this agent has an option (branching

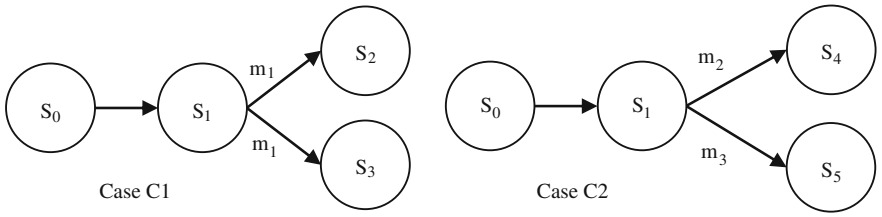


Fig. 4 Branching choices

choice) to follow when it reaches its third state by receiving the message "Analyze". Therefore, the emergent behavior of Fig. 2 can happen.

3.4 System Level Analysis

Visualizing the Agent's Sequence of MSCs.
System level analysis is referred to as detection of implied scenarios, when all components/agents of the system are performing their task and a new scenario is implied to the system. The system behavior should be considered in system level which means following the sequence of all scenarios in the hMSC while there is no central controller for the agents' actions in system level. This problem arises because the agents have local view from the system. In other words, each agent is independent of other agents when completing its actions and the agent has no control on its received messages. Consequently, they are not aware of the sequence of scenarios that other agents are performing. The result can be following different sequences of MSCs by various agents which results to have an implied scenario. Based on Uchitel, Kramer, and Magee's interpretation "*Implied scenarios are the result of specifying the global behaviors of system that will be implemented component wise*" [22]. This explanation is illustrated by an example from Uchitel and et al. paper [22]. The example in Fig. 5 consists of four components interacting in four MSCs. The overall behavior of the system is shown in an hMSC in right side of this figure.

An implied scenario that can happen in this example is shown in Fig. 6 and is taken from [22].

In this scenario, analysis is done based on the previous data that sensor has registered to database. However, the sensor is initiated, terminated, and then initiated again. But database, actuator, and control components analyze the first round of data from sensor component, before it was terminated. This is not acceptable based on MSCs and hMSCs.

For better investigation of this case, let's divide agents into two groups: *Passive* and *Active* agents. *Passive* agents are those agents who start an MSC by receiving a message from other agents. For example components DB and Actuator in our example

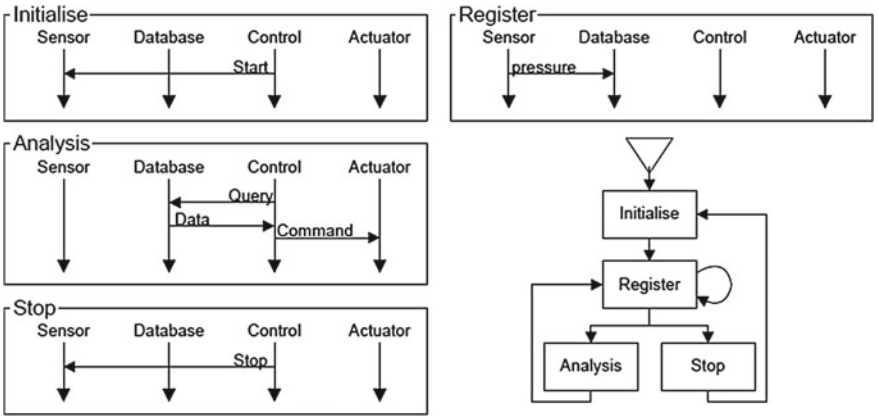


Fig. 5 Four scenarios in the form of MSCs and the associated hMSC (Taken from Uchitel [22])

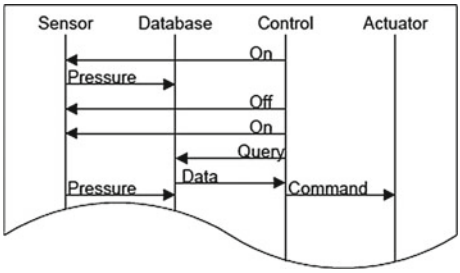


Fig. 6 An implied scenario for system of Fig. 5 (Taken from Uchitel [22])

are *passive* components. *Active* agents like Control component in the example are defined as the agents who start an MSC by sending a message to other agents.

The sequence of MSCs followed by each component in the example is shown in Fig. 7. The name of each component is written above each graph. The abbreviations I, R, A, and T are used for MSC names Initialize, Register, Analysis, and Terminate (Stop) respectively. The dotted arrows show that the component has not involved in the associated MSCs. In other words, this component has not sent or received any messages in these MSCs, and therefore does not follow the paths between these MSCs in the hMSC. As an example, as it is obvious from Fig. 5, the Actuator has no communication with other components in the MSCs other than Analysis scenario. Therefore, there is no path between the MSCs in the Actuator's sequence of MSCs in Fig. 7. These sequences are derived from the MSCs and the hMSC, which can be extracted from the interaction matrices.

Based on what was derived as the sequence of MSCs for each component, the real sequence of MSCs for each component is shown in Fig. 8, where the dotted arrows are omitted. The name of the component is written on the left side of each graph. These graphs show that component Sensor has no involvement in Analysis scenario, DB component does not contribute in Initialize and Terminate MSCs, component

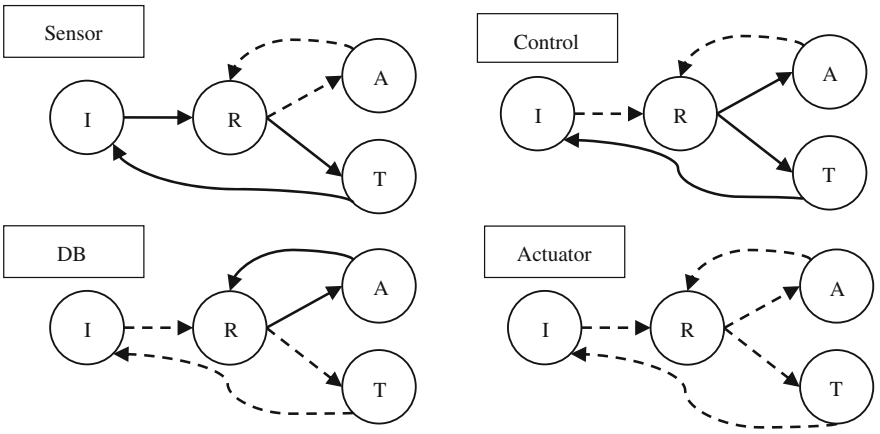


Fig. 7 Sequence of MSCs for components of Fig. 5

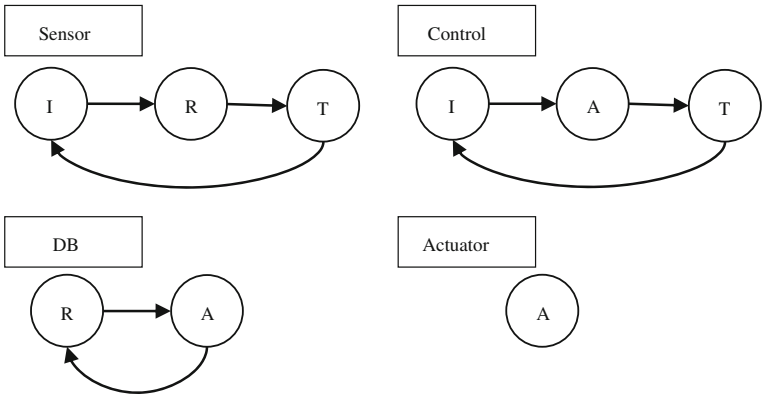


Fig. 8 Sequence of MSCs demonstrating involving MSCs for each of components of Fig. 5

Control does not take part in Register MSC, and Actuator just has communications in Analysis MSC.

The graph of the agent's sequence of MSCs can be interpreted as an abstraction of the MSCs of the system from the agent's point of view. In other words, it shows how the agent realizes the hMSC to perform its actions. From the designer's point of view, the agent's sequence of MSCs can be interpreted as an abstraction of the agent's states in the whole system and in hMSC which shows how the MSCs are performed by that agent.

One possible path in the agent's sequence of MSCs that can lead to an implied scenario is the existence of loops in the internal MSCs of an hMSC. It can affect the system behavior, since the agent may perform the loop more than what is required by other agents, or when others agent are performing the next MSCs, this agent can continue the loop without considering a termination action and the whole system

behavior. DB component is an example of a component which has such a path in its sequence of MSCs.

The other possible path in an Active agent's sequence of MSCs is the existence of a path between internal MSCs of hMSC where the path does not include the initial and termination nodes of hMSC. In this case an implied scenario can exist since the agent is a sender in the startup of the MSCs it is involved in. Therefore, the agent considers the internal MSCs as the initial MSCs and can start at any point in the middle of hMSC without considering the position (hMSC nodes or MSCs) of other agents in the hMSC (i.e. without considering the MSCs that other agents are executing). This can be considered as having a sequence of MSCs like the ones related for DB or Actuator components, however, the components in our example are not Active.

Visualizing the Network of Agents' Interactions.
The other factor other than emergent behavior analysis and implied scenarios that can be investigated by visualization techniques is related to the communication performance. The visualization of the network of agents' interactions can help focusing on analysis of agents with specific characteristics and can lead to a clue for better design of the system. Regarding communication balancing and agents' performance, the agents' interaction network is used to analyze the betweenness and centrality for the agents. Agents with higher centrality/betweenness should be:

1. Analyzed for meeting the performance requirements.
2. Analyzed for emergent behavior analysis. Since the communication between these agents and other agents is high and they play a critical role in the system, any problem that arises with these agents can cause the whole system to fail. On the other hand, the problems in the behavior of these agents can propagate in the system and affect the system behavior.

These two points are especially important for the agents that play a coordinator role in inter-communication between various agent types in MAS.

3.5 Data Preparation

As mentioned earlier, the MSCs and hMSC are the base inputs of the system. These two charts are used to indicate system agents and components involved, the interaction of these components to their environment and among themselves, general behavior of the agents and the whole system, and to what order the scenarios should act (i.e. system behavior).

For preparing our data for further analysis, these MSCs are translated to analyzable data structure, namely interaction matrices. In this technique, the scenarios in the form of MSCs are transformed to send/receive matrices. In synchronous communications (i.e. Messages are considered to be received at the same time they are sent), either send or receive matrices are applicable, because they are transposes of each other. Nevertheless, in asynchronous communications both send and receive matrices

should be utilized. The rest of this paper is based on considering the synchronous communication style. Definitions are followed:

Matrix. One $n \times n$ matrix for each scenario (MSC) is built where n is the number of agents in the system.

Definition 1 (*Send matrix*) Matrix S is defined as the send matrix. $\mathrm{MSC_k}$ is the kth MSC of the system. For each $\mathrm{MSC_k}$ if there is a message sent from agent A_i to agent A_j the order of message as appeared in the scenario is entered as s_{ij} entry of matrix S_k, otherwise $s_{ij} = 0$. If there is more than one message sent from one agent to another, the numbers are separated with a comma in that entry. The S_k is the send matrix related to $\mathrm{MSC_k}$.

In this context each matrix is informative, since it shows the agents, their interactions, the interaction types (i.e. sending or receiving), and the order of messages. The only information missing is the message contents (not message labels).

For component level analysis, the states of each agent should be extracted from the Send matrix of Definition 1. The extracted states should contain information about the transitions between them, the message labels in this transition, and the type of message (Send or Receive). In the finite automata each message can transmit the agent from one state into another. This transition can be extracted from the matrices defined earlier. The state and state transition formal definitions are explained by Alur [6].

Considering the synchronous communication and send matrices, other definitions based on this basic definition can be outlined to help investigate the emergent behavior detection process and show the specific information about agents.

The entities of S matrix in row and column i are the related message numbers for A_i. For each agent A_i in matrix S_k the entities s_{is} show the entities in row i and column s where $1 \leq s \leq n$ and n is the total number of agents in the system. This means the message numbers that agent A_i has sent to other agents and/or to itself. Also the entities s_{ri} show the entities in column i and rows r where $1 \leq r \leq n$. This means the message numbers that agent A_i has received in the related $\mathrm{MSC_k}$. The order of the numbers in the row and column i shows the order of states for that agent. In the case that the agent had sent a message to itself, two states are considered, one for sending and the other for receiving a message. The row number other than i show the agents that send messages to agent A_i.

Definition 2 (*Agent send communication vector*) Vector $v_{isk} = (\mathrm{e_{i1}}, \mathrm{e_{i2}}, \ldots, \mathrm{e_{in}})$ shows the vector of row i in matrix S_k and n is the total number of agents. When there is more than one number in one entry of the S matrix, the element is considered as a compound element. It means for example if there is entry $\mathrm{s_{i1}} = 2, 3$ in matrix S_k; then in the vector v_{isk} element $\mathrm{e_{i1}}$ will be $\mathrm{e_{i1}} = (2, 3)$. This vector shows all of the communications of agent A_i to other agents in the kth MSC where A_i is the sender of the messages.

Definition 3 (*Agent receive communication vector*) Vector $v_{rik} = (\mathrm{e_{1i}}, \mathrm{e_{2i}}, \ldots, \mathrm{e_{ni}})$ shows the vector of column i in matrix S_k and n is the total number of agents. When there is more than one number in one entry of the S matrix, the element is considered

as a compound element. This vector shows all of the communications of agent A_i to other agents in the kth MSC where A_i is the receiver of the messages.

The state vector of each agent A_i in MSC_k is defined as:

Definition 4 (*Agent state vector*) The states of each agent A_i in kth MSC is shown by a vector $states_{ik}$ which is the ordered vector of the entities of v_{isk} and v_{rik} where the zero entities are omitted and the elements are in ascending order. In the case that an agent had sent a message to itself, the same numbers in the v_{isk} and v_{rik} are added distinctly to the states vector. Therefore two states are considered for this agent by sending a message to itself: one for sending and one for receiving the message. The number of the elements in vector $states_{ik}$ shows the total number of events on agent A_i in MSC k or the projection of events of kth MSC on A_i ($\mathrm{MSC_k}|_{\mathrm{Ai}}$). Elements of vector $states_{ik} = (st_1, st_2, \ldots, st_{f-1}, st_f)$ are the states of agent A_i. The transitions between the states are by messages sent or received to/by agent A_i. The last element st_f shows the accepted state of agent A_i in MSC k.

Note that since each element of the state vector $states_{ik}$ matches an element in v_{isk} or v_{rik}, we can trace the source entity s_{kp} in the related S matrix that it matches. Therefore, the type of the message or state can be recognized based on whether it matches a send event or a receive one.

The advantage of using these matrices is that the information about the agent that causes the state transition for each individual agent can be identified by tracking the number of that element in each of the v_{isk} or v_{rik} vectors.

Definition 5 (*State transition sender vector*) The state transition sender vector for agent A_i in MSC k is shown by $Sender_{ik} = (Snd_1, Snd_2, \ldots, Snd_p)$ where each element shows the sender for a message that changes the state for agent A_i. Each element of $states_{ik}$ has a matching either in $v_{isk} = (\mathrm{e_{i1}}, \mathrm{e_{i2}}, \ldots, \mathrm{e_{in}})$ or in $v_{rik} = (\mathrm{r_{1i}}, \mathrm{r_{2i}}, \ldots, \mathrm{r_{ni}})$. Therefore, the senders of the messages for each state are either agent A_i (if the element in $states_{ik}$ has a matching in v_{isk}) or are the agent A_w (if the element in $states_{ik}$ has a matching with the wth element in v_{rik}). Simply explaining, this vector shows which agent is the sender of each event for agent A_i in MSC k.

Greenhouse System Example.
The scenarios in Fig. 1 are a part of scenarios of a greenhouse multiagent system. This system consists of three different types of agents: Temperature balancing (At), Water control (Aw), and Mineral control (Am) agents. The agents receive environmental information from sensors, connect to data and knowledge bases and analyze the information to perform the best task. The agents are supposed to perform autonomously and interact with each other in order to keep the plants in the best situation. The system is designed using MaSE methodology. MSCs are extracted from the sequence diagrams of MaSE and the agent classes (MaSE artifacts) as explained in [26]. One method for temperature balancing is shown in MSC1 in Fig. 9. The $\mathrm{S_t}$ and $\mathrm{S_m}$ show the temperature and mineral sensors respectively and KB stands for knowledge base. MSC2 in Fig. 10 shows the request of $\mathrm{A_m}$ to the water control agent to serve minerals for the plants. For simplicity, just two of the MSCs of the system are shown here.

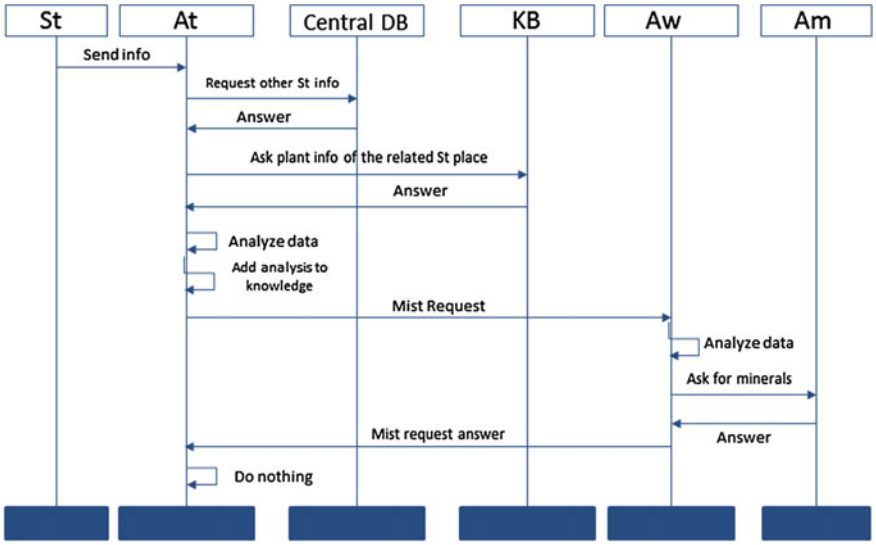


Fig. 9 MSC1: Temperature balancing method

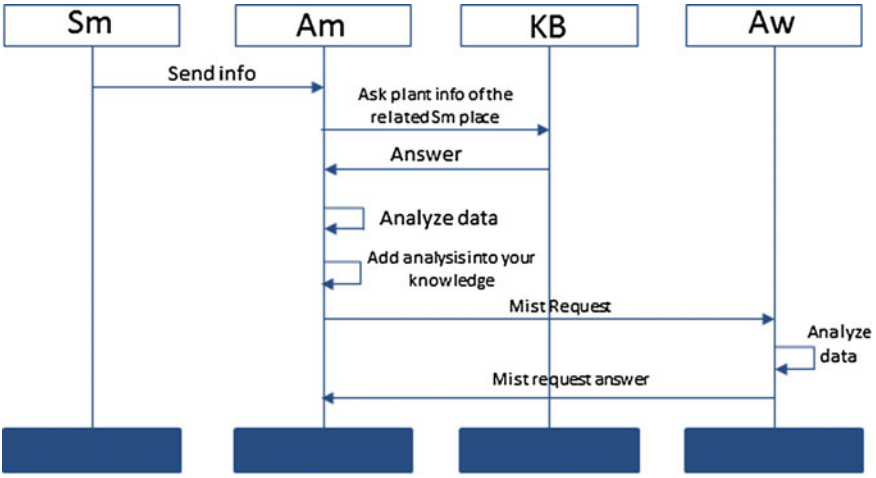


Fig. 10 MSC2 Mineral balancing method

Extract Matrices and Vectors for Emergent Behavior Detection

The first step is to transform the MSCs to related S matrices which show the send matrices of MSCs. As an example, the matrix S_1 related to the MSC1 is shown below:

$$\begin{bmatrix} & S_t & A_t & DB & KB & A_w & A_m & S_m & th \\ S_t & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ A_t & 0 & 6.7 & 2 & 4 & 8 & 0 & 0 & 13 \\ DB & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ KB & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ A_w & 0 & 12 & 0 & 0 & 9 & 10 & 0 & 0 \\ A_m & 0 & 0 & 0 & 0 & 11 & 0 & 0 & 0 \\ S_m & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ th & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

In the MSCs, we will consider the rows and columns ($i = 5$) related to A_w. For this agent in the MSCs, the vectors v_{5sk} and v_{r5k} show the related row and column of A_w and $states_{5k}$ represents the associated states vector of A_w in the related MSCs, where k shows the kth MSC (MSC$_k$). The states vectors of A_w are:

$States_{51} = \{st_1, st_2, st_3, st_4, st_5, st_f\}$
$States_{52} = \{st_1, st_2, st_3, st_f\}$

And the associated message labels to each of the state vectors are:

$mLabels_{51} = \{\mathrm{m}_1, \mathrm{m}_2, \mathrm{m}_2, \mathrm{m}_3, \mathrm{m}_4, \mathrm{m}_5\}$
$mLabels_{52} = \{\mathrm{m}_1, \mathrm{m}_2, \mathrm{m}_2, \mathrm{m}_6\}$

For this agent, the state transition sender vectors are shown with $Sender_{5k}$ in the kth MSC. The sender vectors are:

$Sender_{51} = \{\mathrm{A}_t, \mathrm{A}_w, \mathrm{A}_w, \mathrm{A}_w, \mathrm{A}_m, \mathrm{A}_w\}$
$Sender_{52} = \{\mathrm{A}_m, \mathrm{A}_w, \mathrm{A}_w, \mathrm{A}_w\}$

The state's network of A_w for these two MSCs is shown in Fig. 11 with message labels and senders of each state transition for each of them is shown above them. The initial and final states are differentiated.

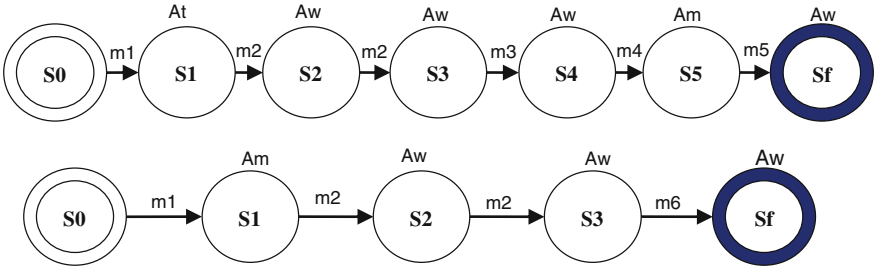


Fig. 11 Agent Aw's states network for MSC1 and MSC2

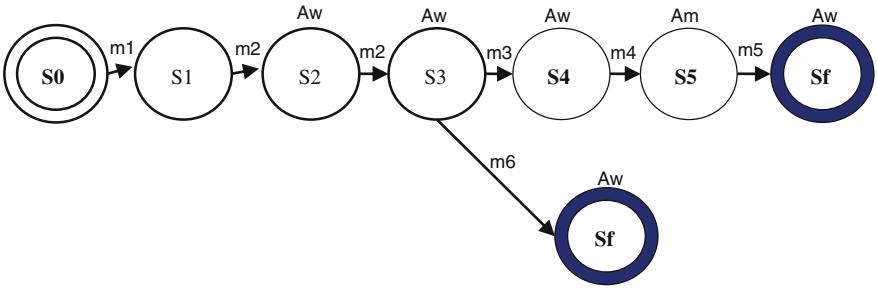


Fig. 12 Visualized states' network of Aw for two MSCs

3.6 Data Visualization

Base of the agent's states network is constructed by one of its state vectors. Other nodes are added to this graph from other agent's state vectors either by adding a separate node when the agent's state is different or by merging to the existing graph when some states are the same with existing nodes. The weights of the edges for the shared parts are increased based on the times of repetition. This can be visualized by bolding the edges or by color changes for edges' weights. Note that shared states can create various branches to the graph. The state transitions or messages are the edge labels for this network.

Continue our example of greenhouse system, the states' network of A_w can form like the one in Fig. 12. Since a part of the states' vectors in the two MSCs are the same, they share part of the graph and then create a branch in the states' network. The shared states and edges are bolded. However, this is not a proof for being same states regarding the states, messages labels, and sender transitions. As it is shown in the figure, state s_1 shows no sender name, because the senders in the two MSCs are different for this state.

For system level and visualization of agents' interaction network, the matrices in Definition 1 are used. The entries of matrices show all of the communications between the agents in one MSC. The number of the messages sent to each agent can be extracted from the same matrix and depends on the associated entry. Other matrices are used later to specify more interactions or change the network based on the number of messages sent to other agents. The weight of edges is defined as the number of messages communicated between two agents.

3.7 Verification

Verifying New Paths Against MSCs.
At component level there are branching choices that visualized in previous sections based on the agents' states. These branches may cause an emergent behavior in the system. However, not all of the branching choices lead the agent to emerge a new behavior. The visualization technique acts as an indication to the possible points of

such behaviors for an agent. Therefore, they should be verified against the MSCs and defined system behavior to detect violations. When we traverse the graph from the initial node and follow the other nodes through connecting edges, by following each branch we will have various paths to reach the final state (last node of the directed graph). These paths can be identified by nodes and edges, states and messaged labels respectively, associated to the state transition senders. These senders can be extracted by the vectors in Definition 5. Therefore, all paths include the senders' information as well.

Verification of each branch is by determining whether or not the path that contains the branch is among the agent's state vectors with respect to both *message labels* and *state transition senders*.

Consider the emergent behavior of greenhouse system shown in Fig. 2. The allowed paths for A_w in MSC1 and MSC2 are:

$$\begin{aligned} Acceptable\_Path_{51} &= \{st_1, m_1, A_t; st_2, m_2, A_w; st_3, m_2, A_w; st_4, m_3, A_w; st_5, m_4, \\ &\qquad A_m; st_6, m_5, A_w\} \\ Acceptable\_Path_{52} &= \{st_1, m_1, A_m; st_2, m_2, A_w; st_3, m_2, A_w; st_4, m_6, A_w\} \end{aligned}$$

The elements are separated by a semicolon, and each element consists of a tuple (*st, m, A*) where *st* shows the state of the agent, *m* stands for associated message label, and the sender transition for that state is shown by *A*. However, the paths that are extracted from the network contain other paths:

$$\begin{aligned} New\_Path_1 &= \{st_1, m_1, A_t; st_2, m_2, A_w; st_3, m_2, A_w; st_4, m_6, A_w\} \\ New\_Path_2 &= \{st_1, m_1, A_m; st_2, m_2, A_w; st_3, m_2, A_w; st_4, m_3, A_w\} \end{aligned}$$

As it was mentioned before, this is a new path which is not verified based on the allowable paths extracted from MSCs. Therefore, an emergent behavior can happen in the system.

Verifying MSC Sequences Against hMSC.
Two possible conditions that can lead to an implied scenario were investigated previously. The first one is the existence of loops in the internal MSCs of an hMSC in the agent's sequence of MSCs. The other condition is the existence of a path between internal MSCs of hMSC in the agent's sequence of MSCs where the path does not include the initial and termination nodes of hMSC for an *Active* agent. These two conditions can simply be detected and verified.

However, there is another condition that leads to an implied scenario and needs to be verified against the hMSC. These can be verified by finding the paths in agent's sequence of MSCs and verifying it against the hMSC. For example, consider the hMSC of Fig. 4. In this example, the MSC sequences extracted from the hMSC and MSCs for control component shows a path of MSCs as follows: Initialize, Analysis, and Terminate, and then from Terminate MSC it can go back to Initialize MSC again. This is shown in Fig. 7. However, when comparing this sequence of MSC {Initialize, Analysis, Terminate} (Fig. 8) there is not such a sequence in hMSC, because there is no direct path from Analysis MSC to Terminate MSC. When verifying the control sequence of MSCs against the hMSC, this contradiction is detected which can lead to another implied scenario in the system.

3.8 Generating Report

The report is generated to inform the designer about the possible emergent behaviors and implied scenarios, the involved agents, the reasons of occurring such behaviors, and the MSCs and states that have conflicts with the MSCs and hMSC of the system. This report helps to announce the problematic points and leads to solve the problem as well.

As an example, consider the greenhouse system. The emergent behavior explained previously is because of the conflicts of the new path with the accepted paths in the MSCs. The new paths contain the same states, message labels, and sender transitions for most of its elements. However, two conflicts are found. The first conflict is in the first element of two new paths $(st_1, \mathrm{m}_1, \mathrm{A}_\mathrm{t})$ or $(st_1, \mathrm{m}_1, \mathrm{A}_\mathrm{m})$ which has a different sender, comparing to similar accepted paths from the system specifications (MSCs). The other conflict is in the message label of the last element of these paths $(st_4, \mathrm{m}_6, \mathrm{A}_\mathrm{w})$ or $(st_4, \mathrm{m}_3, \mathrm{A}_\mathrm{w})$. These two conflicts show that either the senders of first states should be the same for two MSCs or the message labels of the last element should be changed, based on the message labels of the acceptable paths and the shared states of all paths. The report contains all this information which suggests a solution as well. When the changes are defined the new MSC specifications should be checked again to have consistency.

4 Discussion

Social network analysis has various applications. One of the applications of them is shown in this paper through modeling the agents' interaction into various interaction network and analysis of these networks for system verification. The network visualization explained in this paper is used for MAS verification against emergent behavior and implied scenarios. The aim of this work is the detection of these behaviors and whether or not they are accepted or declined depends on the designers and stakeholders. However, these emergent behaviors should be detected earlier in the system to prevent or reduce later damage and costs.

The agent's states' network may seem similar to the state machines. However, they are semantically different from state machines. State machines are not always shown to the designer and the process of behavioral modeling is also different. The alphabet, words, and languages are important in state machines; while in agent's states' network other features are used. Some information about the senders, message types, and transitions of each state are preserved in the network which makes the specification, features, and applications of these networks different from state machines.

One major discussion about emergent behavior detection methodologies is whether they can detect all the unexpected behaviors or not. For our methodology, we are currently working on classification of types of emergent behaviors in these systems. By classification of various scenarios that can happen in a system, we can investigate

this issue, and also suggest solutions to fix these behaviors. The solutions that are suggested and the visualization of the problematic parts are among contributions of this work that are not present in the existing works. As a basic verification of our methodology, until now, all the implied scenarios of other works for instance for the example of Uchitel's case [22] are detected by our methodology. Furthermore, some other issues that can result in an implied scenario are presented which were not discussed in other works. However, in this paper the correctness or completeness of our methodologies is not mentioned.

One of the advantages of our methodology is transforming MSCs to interaction matrices without underlying semantics, which is inspired by social network analysis techniques. This abstraction from the semantics is valuable in terms of removing synthesis phase for emergent behavior detection which is mentioned to be more complex than the model checking. Model checking has problems like scalability for large scale systems. Some proofs for providing scalability by this methodology is applying techniques to detect agents with no emergent behavior in early steps and omit them from further analysis, which helps improve the scalability of the system. The other issue is that matrices have n^2 objects where most of them are zero entities. This is solved by the definitions when extracting appropriate vectors for analysis. The vectors contain just non-zero elements which is much less than n^2. Therefore the method has neither the analysis of n^2 objects nor the zero entities for scalability problem. It is worth mentioning that the analysis for detection of agents with no emergent behavior became feasible by defining interaction matrices rather than using formal method and state machine approaches. This is another advantage of using network analysis in the verification of DSS and MAS.

The other contribution is on visualizing the results which is the main focus of this paper. Based on our knowledge, the three networks discussed in this paper and the visualization techniques for this problem are not indicated in other works. The visualization of these analyses for the network of agents and their states helps the designer for faster fixing of the represented results for detected emergent behaviors. It helps not only see what and where in the system, but also how the emergent behavior is happening. This is another contribution of the work. Most of the existing methodologies detect the emergent behavior without leading the reasons or a solution for it [17]. The generated reports contain all the information about the detected emergent behavior or implied scenario, and therefore covering the shortage of the existing approaches.

The major difference of our work with other researches that try to find emergent behaviors and implied scenarios is using network mining rather than modeling with formal methods. In our work, we try to model the interaction of system components in networks. Then we define the reasons of emerging new behaviors and find criteria and restrictions for the extracted networks. These criteria are used as the main verification method for the extracted networks to investigate and verify them against having emergent behaviors and implied scenarios. Since the modeling is quite different in our work and other researches, the comparison criteria can be quite different. Table 1 represents a comparison between existing main approaches in this field and our work. Some criteria in this table are the level of analysis and whether or not they are capable

Table 1 Comparison of other approaches and our work

Research/ criteria	Source diagram	Modeling	CLA[a]	SLA[b]	Origins of problem[c]	Visualize results	Suggest solution
Uchitel et al. [8, 22, 32]	MSC/ hMSC	Labeled Transition System (LTS)	No	Yes	No	LTS	X
Genest et al. [25][d]	MSC/ hMSC/MSG	State machines	Yes	Yes	X	X	X
Whittle [30]	UML SD	Statechart	Yes	Yes	X	X	X
Mousavi [4]	MSC/ hMSC	Finite state machine	Yes	Yes	X	X	X
Alur et al. [6][e]	MSC/ hMSC/MSG	Automata temporal logic formulas	Yes	Yes	X	X	X
Song et al. [47]	UML SD	Causal graphs	X	Yes	Yes	Orders/ Causal graphs	X[f]
Our work	MSC/ hMSC	Interaction networks/ MSC sequences	Yes	Yes	Yes	Yes	Yes

[a]CLA is used as an abbreviation for Component Level Analysis
[b]SLA is used as an abbreviation for System Level Analysis
[c]This criteria means that the source of the problem is detected and shown to the designer rather than just the existence of an emergent behavior is notified. This is due to the exact cause of emergent behavior both in system and component level analysis, as mentioned in literature, is the local view or restricted view of the components of the system
[d]This paper and their recent papers of the authors mostly discuss the MSC/hMSC specification languages and the validation problems about model-checking and implementability
[e]This paper and related papers of the authors mostly discuss on the computations and validation features of MSC/hMSC/MSG. They discuss the decidability and implementability as well as complexity of model checking
[f]They just provide the parts of the system that should be considered to handle the implied scenarios

of visualizing the analysis results in an effective way to the designer (i.e. in a way that leads to a solution or revising the designs).

5 Conclusion and Future Work

Taking advantage of visualization techniques is an important factor in verification of MAS for detection of emergent behaviors and implied scenarios. The networks defined and presented in previous sections prove the role of this technique in the verification field. These networks are used both for component level and system

level analysis and demonstrates the wide application of network visualization in MAS verification. Although the definitions are verified on MAS, the methodology and visualization technique can be used for verification of DSS as well. Since in this approach we mostly deal with the behavior of the system in terms of MSCs and hMSC and not the internal knowledge or behavior of agents. Consequently, the terms used in this article can be used for DSS verification as well.

Two main future lines of works are in our plans. First, demonstrating various elements such as message types, type of states, and whether the agent is a sender after branching choice in its states' network that can be added as visualization options to the system. Second, bolding the shared states and color differentiation among the shared states with various sender transitions can be illustrated in the network. These features make the designer aware of possible problems in advance before analyzing and verifying the whole system. Other possible future directions include considering other measurements for nodes and links and investigating the effectiveness of those features in the software verification problem.

Acknowledgments This research is supported by a grant from Izaak Walton Killam Memorial Scholarship, Alberta Innovates Technology Futures and partially from Natural Sciences and Engineering Research Council of Canada.

References

1. Wooldridge M, Fisher M, Huget M-H, Parsons S (2002) Model checking multi-agent systems with MABLE. In: Proceedings of the first international joint conference on autonomous agents and multiagent systems: part 2. ACM: Bologna, pp 952–959
2. Bordini R, Fisher M, Pardavila C, Visser W, Wooldridge M (2003) Model checking multi-agent programs with CASP. In: Hunt W Jr, Somenzi F (eds) Computer aided verification. Springer, Berlin, pp 110–113
3. Moshirpour M, Mousavi A, Far BH (2010) A technique and a tool to detect emergent behavior of distributed systems using scenario-based specifications. In: 22nd IEEE international conference on tools with artificial intelligence (ICTAI)
4. Mousavi A (2009) Inference of emergent behaviours of scenario-based specifications. In: Department of electrical and computer engineering. University of Calgary, Calgary, p 160
5. Broy M (2011) Seamless method- and model-based software and systems engineering. The future of software engineering. Springer, Berlin, pp 33–47
6. Alur R, Etessami K, Yannakakis M (2003) Inference of message sequence charts. IEEE Trans Softw Eng 29(7):623–633
7. Chakraborty J, D'Souza D, Narayan Kumar K (2010) Analysing message sequence graph specifications. In: Margaria T, Steffen B (eds) Leveraging applications of formal methods, verification, and validation. Springer, Berlin, pp 549–563
8. Uchitel S (2003) Incremental elaboration of scenario-based specifications and behaviour models using implied scenarios. In: Department of computing. Imperial College of Science, Technology and Medicine, University of London, p 173
9. Chaki S, Clarke E, Grumberg O, Ouaknine J, Sharygina N, Touili T, Veith H (2005) State/event software verification for branching-time specifications. In: Romijn J, Smith G, Pol J (eds) Integrated formal methods. Springer, Berlin, pp 53–69
10. Sanchez E, Squillero G, Tonda A (2012) Automatic software verification. Industrial applications of evolutionary algorithms. Springer, Berlin, pp 17–30

11. Ammar K, Pullum L, Taylor B (2006) Augmentation of current verification and validation practices. Methods and procedures for the verification and validation of artificial neural networks. Springer, New York, pp 13–31
12. Quan TT, Hoang DLN, Nguyen BT, Nguyen AN, Tran QD, Nguyen PH, Bui TH, Do AT, Huynh LV, Doan NT, Huynh NT, Nguyen TD, Nguyen TT, Nguyen VH et al (2010) MAFSE: a model-based framework for software verification, pp 150–156
13. Verhulst E, Jong G, Mezhuyev V (2008) An industrial case: pitfalls and benefits of applying formal methods to the development of a network-centric RTOS. In: Cuellar J, Maibaum T, Sere K (eds) FM 2008: formal methods. Springer, Berlin, pp 411–418
14. Briand LC (2010) Software verification—a scalable, model-driven, empirically grounded approach. Simula Research Laboratory. Springer, Berlin, pp 415–442
15. Knight J (1998) Challenges in the utilization of formal methods. In: Ravn A, Rischel H (eds) Formal techniques in real-time and fault-tolerant systems. Springer, Berlin, pp 1–17
16. Kneuper R (1997) Limits of formal methods. Form Asp Comput 9(4):379–394
17. Song I-G, Sang-Uk J, Ah-Rim H, Doo-Hwan B (2011) An approach to identifying causes of implied scenarios using unenforceable orders. Inf Softw Technol 53(6):666–681
18. Bontemps Y, Schobbens P-Y (2007) The computational complexity of scenario-based agent verification and design. J Appl Log 5(2):252–276
19. Fard FH, Far BH (2013) Detection and verification of a new type of emergent behavior in multiagent systems. In: 17th international conference on intelligent engineering systems (INES)
20. Fard FH (2013) Detecting and fixing emergent behaviors in distributed software systems using a message content independent method. In: Press in 28th IEEE/ACM international conference on automated software engineering (ASE), Doctoral symposium. IEEE
21. Fard FH, Far BH (2012) A method for detecting agents that will not cause emergent behavior in agent based systems—a case study in agent based auction systems. In: 2012 IEEE 13th international conference on information reuse and integration (IRI)
22. Uchitel S, Kramer J, Magee J (2002) Implied scenario detection in the presence of behaviour constraints. Electron Notes Theor Comput Sci 65(7):65–84
23. Carrol JM (1999) Five reasons for scenario-based design. In: Proceedings of the 32nd annual Hawaii international conference on system sciences, HICSS-32
24. Lunjin L, Dae-Kyoo K (2011) Required behavior of sequence diagrams: semantics and refinement. In: 2011 16th IEEE international conference on engineering of complex computer systems (ICECCS)
25. Genest B, Muscholl A (2005) Message sequence charts: a survey. In: Fifth international conference on application of concurrency to system design, ACSD 2005
26. Mani N, Garousi V, Far BH (2008) Monitoring multi-agent systems for deadlock detection based on UML models. In: Canadian conference on electrical and computer engineering, CCECE 2008
27. Genest B, Muscholl A, Peled D (2004) Message sequence charts. In: Desel J, Reisig W, Rozenberg G (eds) Lectures on concurrency and petri nets. Springer, Berlin, pp 103–121
28. Broy M (2000) The essence of message sequence charts. In: Proceedings of the international symposium on multimedia software engineering 2000
29. Krüger IH (2000) Distributed system design with message sequence charts. In: Institute of computer science 2000, Technical University of Munich, p 386
30. Whittle J, Schumann J (2006) Scenario-based engineering of multi-agent systems in agent technology from a formal perspective. In: Rouff C et al (eds) Springer, London, pp 159–189
31. UNION-, TSSOI-IT, SERIES Z (2004) Languages and general software aspects for telecommunication systems—formal description techniques (FDT)—message sequence chart, ITU-T Recommendation Z.120, p 136
32. Uchitel S, Kramer J, Magee J (2001) Detecting implied scenarios in message sequence chart specifications. In: Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on foundations of software engineering 2001. ACM, Vienna, pp 74–82

33. Alur R, Yannakakis M (1999) Model checking of message sequence charts. In: Baeten JM, Mauw S (eds) Concurrency theory CONCUR'99. Springer, Berlin, pp 114–129
34. Moshirpour M (2011) Model-based detection of emergent behavior in distributed and multi-agent systems from component level perspective. In: Department of electrical and computer engineering 2011, University of Calgary, Calgary, p 109
35. Muccini H (2003) Detecting implied scenarios analyzing non-local branching choices. In: Pezzè M (ed) Fundamental approaches to software engineering. Springer, Berlin, pp 372–386
36. Sousa FCd, Mendonca NC, Uchitel S, Kramer J (2007) Detecting implied scenarios from execution traces. In: Proceedings of the 14th working conference on reverse engineering 2007. IEEE Computer Society, pp 50–59
37. Uchitel S, Kramer J, Magee J (2003) Synthesis of behavioral models from scenarios. IEEE Trans Softw Eng 29(2):99–115
38. Adsul B, Mukund M, Kumar KN, Narayanan V (2005) Causal closure for MSC languages. In: Sarukkai S, Sen S (eds) FSTTCS 2005: foundations of software technology and theoretical computer science. Springer, Berlin, pp 335–347
39. Bhateja P, Gastin P, Mukund M, Kumar KN (2007) Local testing of message sequence charts is difficult, in fundamentals of computation theory. In: Csuhaj-Varjú E, Ésik Z (eds) Springer, Berlin, pp 76–87
40. Alur R, Etessami K, Yannakakis M (2005) Realizability and verification of MSC graphs. Theor Comput Sci 331(1):97–114
41. Alur R, Yannakakis M (2001) Model checking of hierarchical state machines. ACM Trans Program Lang Syst 23(3):273–303
42. Schumann J, Whittle J (2001) Automatic synthesis of agent designs in UML. In: Rash J et al (eds) Formal approaches to agent-based systems. Springer, Berlin, pp 148–162
43. Whittle J, Kwan R, Saboo J (2005) From scenarios to code: an air traffic control case study. Softw Syst Model 4(1):71–93
44. Whittle J, Schumann J (2000) Generating statechart designs from scenarios. In: Proceedings of the 22nd international conference on software engineering 2000. ACM, Limerick, pp 314–323
45. Ben-Abdallah H, Leue S (1998) MESA: Support for scenario-based design of concurrent systems. In: Steffen B (ed) Tools and algorithms for the construction and analysis of systems. Springer, Berlin, pp 118–135
46. Ben-Abdallah H, Leue S (1997) Syntactic detection of process divergence and non-local choice in message sequence charts. In: Brinksma E (ed) Tools and algorithms for the construction and analysis of systems. Springer, Berlin, pp 259–274
47. Song I-G, Jeon SU, Bae DH (2009) A graph based approach to detecting causes of implied scenarios under the asynchronous and synchronous communication styles. In: Proceedings of the 16th Asia-Pacific software engineering conference 2009. IEEE Computer Society, pp 53–60
48. Uchitel S, Kramer J (2001) A workbench for synthesising behaviour models from scenarios. In: Proceedings of the 23rd international conference on software engineering 2001. IEEE Computer Society, Toronto, Ontario, pp 188–197
49. Letier E, Kramer J, Magee J, Uchitel S (2008) Deriving event-based transition systems from goal-oriented requirements models. Autom Softw Eng 15(2):175–206
50. Letier E, Kramer J, Magee J, Uchitel S (2005) Monitoring and control in scenario-based requirements analysis. In: Proceedings of the 27th international conference on software engineering 2005. ACM, St. Louis, pp 382–391
51. Magee J, Pryce N, Giannakopoulou D, Kramer J (2000) Graphical animation of behavior models. In Proceedings of the 22nd international conference on software engineering 2000, ACM, Limerick, pp 499–508
52. Henriksen J, Mukund M, Kumar KN, Thiagarajan PS (2000) Regular collections of message sequence charts. In: Nielsen M, Rovan B (eds) Mathematical foundations of computer science 2000. Springer, Berlin, pp 405–414
53. Mukund M, Kumar KN, Sohoni M (2000) Synthesizing distributed finite-state systems from MSCs. In: Palamidessi C (ed) CONCUR 2000—concurrency theory. Springer, Berlin, pp 521–535

54. Aggarwal C (2011) An introduction to social network data analytics. In: Aggarwal CC (ed) Social network data analytics. Springer, New York, pp 1–15
55. Sabater J, Sierra C (2002) Reputation and social network analysis in multi-agent systems. In: Proceedings of the first international joint conference on autonomous agents and multiagent systems: part 1. ACM, Bologna, pp 475–482
56. Hanneman RA, Riddle M (2005) Introduction to social network methods. http://faculty.ucr.edu/~hanneman/nettext/C6_Working_with_data.html
57. Mislove AE (2009) Online social networks: measurement, analysis, and applications to distributed information systems. Rice University
58. Song HH, Cho TW, Dave V, Zhang Y, Qiuet L (2009) Scalable proximity estimation and link prediction in online social networks. In: Proceedings of the 9th ACM SIGCOMM conference on internet measurement conference 2009. ACM, Chicago, pp 322–335
59. Sun J, Tang J (2011) A survey of models and algorithms for social influence analysis. In: Aggarwal CC (ed) Social network data analytics. Springer, New York, pp 177–214
60. Gutiérrez C, García-Magariño I, Fuentes-Fernández R (2011) Detection of undesirable communication patterns in multi-agent systems. Eng Appl Artif Intell 24(1):103–116
61. Gutierrez C, García-Magariño I (2011) Revealing bullying patterns in multi-agent systems. J Syst Softw 84(9):1563–1575
62. Gutiérez C, García-Magariño I, Gómez-Sanz J (2009) Evaluation of multi-agent system communication in INGENIAS. In: Cabestany J et al (eds) Bio-Inspired systems: computational and ambient intelligence. Springer, Berlin, pp 619–626
63. Botía J, Hernansáez J, Gómez-Skarmeta A (2007) On the application of clustering techniques to support debugging large-scale multi-agent systems. In: Bordini R et al (eds) Programming multi-agent systems. Springer, Berlin, pp 217–227
64. Botía J, Gómez-Sanz J, Pavón J (2006) Intelligent data analysis for the verification of multi-agent systems interactions. In: Corchado E et al (eds) Intelligent data engineering and automated learning—IDEAL'06. Springer, Berlin, pp 1207–1214

Glossary

Blog A blog is a website that is maintained by an individual or a small group and is updated regularly.

Facebook Facebook is a social network that allows its users to share information in various media types.

Friend Friends are users who connect with each other and share information in a social network.

Gender Prediction Gender prediction is the process of predicting the gender of a social network user from publically available indicators, such as the user's posts or the colors used in his/her profile.

Genetic Algorithm A genetic algorithm is an optimization algorithm that is inspired by natural selection in Biology. It employs biological operators such as cross-over and mutation in order to find a near optimal solution.

Hashtag A hashtag is metadata and additional context added to tweets; they start with the # symbol.

Microblog A microblog is a blog with short and frequent posts.

Social Media Social media are user created information that are published and shared with others in social networks or other online platforms. This information can be in many formats including text, video, or audio.

Social Network A social network is an online application that allows users to communicate with each other by sharing information in various forms.

Social Network Analysis Social network analysis is the study of social networks utilizing network/graph theory.

Social Networking Social networking is the act of interacting with other users of a social network.

Social Visualization Social visualization is the visualization of social data for social purposes. Its focus is users and the patterns of their interactions.

J. Kawash (ed.), *Online Social Media Analysis and Visualization*,
Lecture Notes in Social Networks, DOI 10.1007/978-3-319-13590-8

Trend Prediction A trend in social networks is when information noticeably propagates through the network. Trend prediction refers to the process of predicting the outcomes of a trend.

Tweet A tweet is a 140-character message that users share on Twitter.

Twitter Twitter is a social network that allows its users to share short 140-character messages, called tweets.

Visualization Visualization is the graphical depiction of information; it emphasises the relationships between various depicted objects, facilitating the formation of a mental image.

Web Mining Web mining applies data mining techniques to the Web; it typically analyzes content, usage, and structure of the Web.

Word-of-Mouth Marketing Word-of-Mouth Marketing refers to the sharing of information between social network users about a product or service.

Index

J. Kawash (ed.), *Online Social Media Analysis and Visualization*,
Lecture Notes in Social Networks, DOI 10.1007/978-3-319-13590-8