Martin Atzmueller
Andreas Hotho
Markus Strohmaier
Alvin Chin (Eds.)

LNAI 6904

Analysis of Social Media and Ubiquitous Data

International Workshops
MSM 2010, Toronto, ON, Canada, June 2010, and
MUSE 2010, Barcelona, Spain, September 2010
Revised Selected Papers

Springer

Martin Atzmueller   Andreas Hotho
Markus Strohmaier   Alvin Chin (Eds.)

Analysis of Social Media and Ubiquitous Data

International Workshops
MSM 2010, Toronto, ON, Canada, June 13, 2010, and
MUSE 2010, Barcelona, Spain, September 20, 2010
Revised Selected Papers

Springer

Series Editors

Randy Goebel, University of Alberta, Edmonton, Canada
Jörg Siekmann, University of Saarland, Saarbrücken, Germany
Wolfgang Wahlster, DFKI and University of Saarland, Saarbrücken, Germany

Volume Editors

Martin Atzmueller
University of Kassel, Germany
E-mail: atzmueller@cs.uni-kassel.de

Andreas Hotho
University of Würzburg, Germany
E-mail: hotho@informatik.uni-wuerzburg.de

Markus Strohmaier
Graz University of Technology, Austria
E-mail: markus.strohmaier@tugraz.at

Alvin Chin
Nokia Research Center, Beijing, China
E-mail: alvin.chin@nokia.com

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-23598-6 e-ISBN 978-3-642-23599-3
DOI 10.1007/978-3-642-23599-3
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011934915

CR Subject Classification (1998): I.2, H.5.3, H.5, C.2, K.4, K.5, K.6

LNCS Sublibrary: SL 7 – Artificial Intelligence

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

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

Preface

In the last decade, the reach of computational systems has dramatically expanded both in breadth and depth. This development has led computational devices and applications to permeate societal and social systems in an unprecedented manner.

Today, an increasing entwinement of social phenomena, ubiquitous data, and computational processes can be observed in many domains and contexts, including social media, online social networking, and mobile computing. Such systems, in which social, ubiquitous, and computational processes are interdependent and tightly interwoven, can be characterized as distributed social – computational systems, i.e., integrated systems of people, sensors and computers. Typically, the properties of such systems can be considered to be emergent, which means (a) they are influenced by a combination of social phenomena, algorithmic computation, and ubiquitous data and (b) they are usually beyond the direct control of system engineers. In such systems, potentially critical system properties emerge through social adoption and usage.

Therefore, understanding and engineering social – computational systems requires novel approaches and new techniques to system analysis and engineering. This book sets out to explore this emerging space by presenting a number of current approaches and early important work addressing selected aspects of this problem. The individual contributions of this book represent the first steps in this direction, focusing on problems related to the modeling and mining of social and ubiquitous computational systems. Methods for mining, modeling, and development can help to advance our understanding of the dynamics and structures inherent to these systems, and can help to make social – computational systems and ubiquitous data amenable to deeper quantitative analysis.

The papers presented in this book are revised and significantly extended versions of papers submitted to two related workshops: The Modeling Social Media Workshop (MSM 2010) that was held on June 13, 2010 in conjunction with the 21st ACM Conference on Hypertext and Hypermedia (Hypertext 2010), in Toronto, Canada, and the Mining Ubiquitous and Social Environments (MUSE 2010) International Workshop, which was held on September 20, 2010 in conjunction with the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD 2010) in Barcelona, Spain. In the following, we briefly discuss the themes of those two workshops in more detail.

Social Media: Social media applications such as blogs, microblogs, wikis, news aggregation sites, and social tagging systems have pervaded the Web and have transformed the way people communicate and interact with each other online. In order to understand and effectively design social media systems, we need to develop models that are capable of reflecting their complex, multi-faceted socio-technological nature. Modeling social media applications enables us to understand and predict their evolution, explain their dynamics, or to describe their underlying social – computational mechanics.

Ubiquitous Data: Ubiquitous data require novel analysis methods including new methods for data mining and machine learning. Unlike in traditional data mining scenarios, data do not emerge from a small number of (heterogeneous) data sources, but potentially from hundreds to millions of different sources. As there is only minimal coordination, these sources can overlap or diverge in any possible way. In typical ubiquitous settings, the mining system can be implemented inside the small devices and sometimes on central servers, for real-time applications, similar to common mining approaches. Steps into this new and exciting application area are the analysis of the collected new data, the adaptation of well-known data mining and machine learning algorithms, and finally the development of new algorithms. The advancement of such algorithms and their application in social and ubiquitous settings is one of the core contributions of this book.

Considering these two workshop themes, the papers contained in this volume form a starting point for bridging the gap between both worlds: Both social media applications and ubiquitous systems benefit from modeling aspects, either at the system level, or for providing a sound data basis for further analysis and mining options. On the other hand, data analysis and data mining can provide novel insights and thus similarly enhance and support modeling prospects. In "A Framework for Mobile User Activity Logging," Wolfgang Woerndl, Alexander Manhardt, Florian Schulze, and Vivian Prinz provide a unified approach for collecting user activity data on mobile devices for user modeling in social computational and ubiquitous systems. In "Intentional Modeling of Social Media Design Knowledge for Government – Citizen Communication," Andrew Hilts and Eric Yu present how the agent-oriented modeling framework i* can be applied to analyze the impact of different social media configurations on the goals and relationships of the actors involved. In "Grooming Analysis—Modeling the Social Dynamics of Online Discussion Groups," Else Nygren presents results from an empirical study of social interactions (in particular: grooming) in a social – computational system.

Next, the chapter "Exploring Gender Differences in Member Profiles of an Online Dating Site Across 35 Countries," Slava Kisilevich and Mark Last describe the construction of classification models for characterizing gender differences in social networking sites, specifically online dating sites for different countries stressing both modeling and mining aspects. In "Community Assessment Using Evidence Networks," Folke Mitzlaff, Martin Atzmueller, Dominik Benz, Andreas Hotho, and Gerd Stumme present a community assessment approach using evidence networks of user activities; the experiments show that (implicit) evidence networks are well suited for consistent relative community ratings, for evaluation and comparison of mined community structures.

The work "Towards Adjusting Mobile Devices to User's Behaviour" by Peter Fricke, Felix Jungermann, Katharina Morik, Nico Piatkowski, Olaf Spinczyk, Marco Stolpe, and Jochen Streicher discusses the optimization of mobile (and ubiquitous) devices with respect to the behavior of users. The paper "Bayesian Networks to Predict Data Mining Algorithm Behavior in Ubiquitous Environments" by Aysegul Cayci, Santiago Eibe, Yucel Saygin, and Ernestina Menasalvas describes an approach for parameter estimation and method adaptation in the context of ubiquitous environments with limited resources. Finally, the paper "Online and Offline Trend Cluster Discovery in Spatially Distributed Data Streams" by Anna Ciampi, Annalisa Appice, and Donato Malerba discusses an algorithm for interleaving spatial clustering and trend discovery, with a broad application scope.

It is the hope of the editors that this book (a) catches the attention of an audience interested in recent problems and advancements in the fields of social media, online social networks, and ubiquitous data and (b) helps to spark a conversation on new problems related to the design and analysis of ubiquitous social – computational systems.

We want to thank our reviewers for their careful help in selecting and improving the provided submissions.

June 2011

Martin Atzmueller
Andreas Hotho
Markus Strohmaier
Alvin Chin

Organization

Program Committee

Martin Atzmueller	University of Kassel, Germany
Ulf Brefeld	Yahoo! Research, Spain
Jordi Cabot	INRIA-École des Mines de Nantes, France
Alvin Chin	Nokia Research Center, China
Marco De Gemmis	University of Bari, Italy
Wai-Tat Fu	University of Illinois, USA
Daniel Gayo Avello	University of Oviedo, Spain
Tad Hogg	Hewlett Packard Laboratories, USA
Andreas Hotho	University of Würzburg, Germany
Thomas Kannampallil	University of Texas, USA
Katharina Morik	TU Dortmund, Germany
Ion Muslea	Language Weaver, Inc., USA
Harald Sack	Hasso-Plattner-Institut, Universität Potsdam, Germany
Sergej Sizov	University of Koblenz-Landau, Germany
Marc Smith	Connected Action Consulting Group, USA
Markus Strohmaier	Graz University of Technology, Austria

Additional Reviewer

Hercher, Johannes

Table of Contents

Logging User Activities and Sensor Data on Mobile Devices 1
Wolfgang Woerndl, Alexander Manhardt, Florian Schulze, and Vivian Prinz

Intentional Modeling of Social Media Design Knowledge for Government-Citizen Communication 20
Andrew Hilts and Eric Yu

Grooming Analysis Modeling the Social Interactions of Online Discussion Groups 37
Else Nygren

Exploring Gender Differences in Member Profiles of an Online Dating Site Across 35 Countries 57
Slava Kisilevich and Mark Last

Community Assessment Using Evidence Networks 79
Folke Mitzlaff, Martin Atzmueller, Dominik Benz, Andreas Hotho, and Gerd Stumme

Towards Adjusting Mobile Devices to User's Behaviour 99
Peter Fricke, Felix Jungermann, Katharina Morik, Nico Piatkowski, Olaf Spinczyk, Marco Stolpe, and Jochen Streicher

Bayesian Networks to Predict Data Mining Algorithm Behavior in Ubiquitous Computing Environments 119
Aysegul Cayci, Santiago Eibe, Ernestina Menasalvas, and Yucel Saygin

Online and Offline Trend Cluster Discovery in Spatially Distributed Data Streams 142
Anna Ciampi, Annalisa Appice, and Donato Malerba

Author Index 163

Logging User Activities and Sensor Data on Mobile Devices

Wolfgang Woerndl, Alexander Manhardt, Florian Schulze, and Vivian Prinz

TU Muenchen, Chair for Applied Informatics / Cooperative Systems (AICOS)
Boltzmannstr. 3, 85748 Garching, Germany
{woerndl,manhardt,schulze,prinzv}@in.tum.de

Abstract. The goal of this work is a unified approach for collecting data about user actions on mobile devices in an appropriate granularity for user modeling. To fulfill this goal, we have designed and implemented a framework for mobile user activity logging on Windows Mobile PDAs based on the MyExperience project. We have extended this system with hardware and software sensors to monitor phone calls, messaging, peripheral devices, media players, GPS sensors, networking, personal information management, web browsing, system behavior and applications usage. It is possible to detect when, at which location and how a user employs an application or accesses certain information, for example. To evaluate our framework, we applied it in several usage scenarios. We were able to validate that our framework is able to collect meaningful information about the user. We also outline preliminary work on analyzing the logged data sets.

Keywords: user modeling, mobile, activity logging, personal digital assistant, sensors.

1 Introduction

Mobile devices like Smartphones and personal digital assistants (PDAs) are becoming more and more powerful and are increasingly used for tasks such as searching and browsing Web pages, or managing personal information. However, mobile information access still suffers from limited resources regarding input capabilities, displays, network bandwidth etc. Therefore, it is desirable to tailor information access on mobile devices to data that has been collected and derived about the user. This information is called the *user model*.

When adapting information access, systems often apply a general user modeling process [1]. Thereby, we can identify three main steps (Fig. 1): 1. Collecting data about the user, 2. Analyzing the data to build a user model, 3. Using the user model to adapt information access.

In this work, we focus on the first step of this user modeling process: the collection of data about the user in a mobile environment. The goal of this work is a unified approach for recording user actions on mobile devices in granularity appropriate for user modeling. To fulfill this goal, we have designed and implemented a framework for mobile user activity logging on Windows Mobile PDAs. The framework handles different kinds of hardware and software sensors in a combined and consistent way.

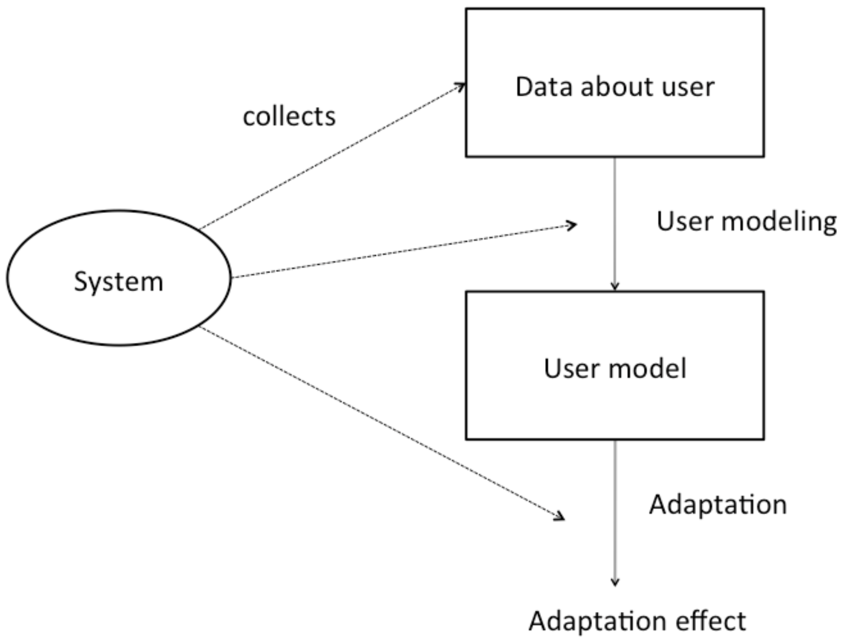


Fig. 1. User modeling process [1]

The remainder of this paper is organized as follows. The next section describes requirements and related work. In Section 3 we explain the design and implementation of our framework for mobile user activity logging. Section 4 covers the evaluation of our approach. We then outline preliminary work on analyzing the logged data sets. Finally, we give a summary and outlook for future work in Section 6.

2 Requirements and Related Work

2.1 Requirements

The most important feature of our mobile user activity logger is to cover all user actions that can occur on a mobile device with associated sensor data. Since the goal of this work is collecting data for a specific purpose (user modeling), it is important to consider the *granularity* of the data recording. To test the usability of a mobile software application, for example, it may be necessary to record movements on a touch screen, single keystrokes or exactly where a user hits a button. This may lead to too much data that has to be handled and stored. On the other hand, if a system only records that a user has been starting the mobile web browser, for example, this information may not be sufficient to be able to derive knowledge about what the user is interested in. For our purpose of user modeling, it is useful to also collect which web sites the user has visited or which keywords she has entered for a web search, for instance.

For hardware sensors, an activity logging system shall record data when user actions lead to a change in the situation the user is in. For example, the system should log when a user is driving or walking around and thus changing her position. Alternatively, the system could record a snap shot of the sensor status at fixed time intervals. This may lead to a lot of redundant data and is not preferable since resources such as storage capacity are limited on the mobile device.

Another focal point to consider is *implicit* versus *explicit* user profile acquisition. Due to the limitation of the mobile user interface, necessary user interactions should be kept at a minimum. Users do not like to fill out forms or answer questions on a mobile device. In addition, the system should take into consideration the mobile-users' limited attention span while moving, changing locations and contexts, and expectations of quick and easy interactions [2]. Therefore, the data collection should be based on observing the user in her ongoing activities without distracting her too much. It is desirable to collect real usage data as it occurs in its natural setting [3]. Explicit user interaction could be optionally used to augment the implicitly collected data from hardware and software sensors. For example, the system could optionally ask whether a user is in a "work" or "leisure" setting in a particular location. By doing so, the user modeling system could later aggregate information from different "leisure" situations.

Finally, every system that collects data about the user has to consider users' *privacy* concerns. For mobile user modeling this is especially important since additional information such as the user position is available. Sensor data may be even more sensible than information users provide in a web form. Therefore, it is desirable to keep the collected data on the mobile device and not send it to a server over a network. By doing so, the user can always shut down the data recording or delete the data. She thus is able to retain control over the collected data. In addition, the user should have an option to manually disable individual sensors. This option is also beneficial to be able to save battery power. An example is to disable the GPS sensor when a user is inside a building for a whole day.

2.2 Related Work

In a nutshell, existing related work is either focused on gathering data in a non-mobile desktop setting, do collect data from specific sensors only (e.g. analysis of user location based on GPS logs) or were created for different purposes other than user modeling.

An example for activity logging in a desktop setting is the approach by Chernov et.al. [4]. Similar to our aim, their goal is to collect data sets about user behavior using a single methodology and a common set of tools. One of their main considerations is to protect the data from unauthorized access. Because all the data is stored directly on the user's computer, it is up to the user to decide to whom and in what form the data should be released. However, it is not available and usable for mobile devices.

There is plenty of work in capturing and analyzing user movement using GPS and other positioning technologies. An example is the Geolife project [5]. The goal is to mine interesting locations and classical travel sequences in a given geospatial region based on GPS trajectories of multiple users. Their model infers the interest of a location by taking several factors into account. However, this work and other similar approaches in mobile user modeling only focus on single or a few sensors such as GPS and do not attempt to record all user actions on mobile devices.

The Mobile Sensing Platform described in [6] is an interesting system designed for embedded activity recognition. It incorporates multimodal sensing, data processing and inference, storage, all-day battery life, and wireless connectivity into a single wearable unit. However, it is an extra device the user has to carry, and the system cannot capture all the everyday activities users perform on their PDAs. [7] is another example of a system that is relying on an external device to record and analyze user activities.

MobSens is a system to derive sensing modalities on smart mobile phones [8]. The authors discuss experiences and lessons learned from deploying four mobile sensing applications on off-the-shelf mobile phones in the framework that contains elements of health, social, and environmental sensing at both individual and community levels. However, the system's focus is on hardware sensors. Actions that users perform with software on a mobile device are not integrated.

MyExperience is an interesting project as it allows for capturing both objective and subjective *in situ* data on mobile computing activities [3]. The purpose of MyExperience is to understand how people use and experience mobile technology to be able to optimize the design of mobile applications, for example. Hence the system is not tailored towards user modeling, but it can serve as a foundation for our implementation, since the framework is extensible. We will therefore describe the MyExperience project in more detail in the next section.

3 Design and Implementation of the Mobile User Activity Logger

In this section we discuss issues concerning the design and implementation of our mobile user activity logger including the various hardware and software sensors. The logger is based on the MyExperience framework.

3.1 The MyExperience Project

MyExperience is a software tool for Windows Mobile PDAs and smartphones based on the Microsoft .NET Compact Framework 2.0 and the Microsoft SQL Compact Edition database. The software is available as a BSD-licensed open source project [9]. MyExperience runs continuously with minimal impact on people's personal devices. It has an event-driven, "Sensor-Trigger-Action" architecture that efficiently processes a variety of sensed events [3]. The collected data is enhanced by direct user feedback to enable capturing both objective and subjective information about user actions.

MyExperience is based on a three-tier architecture of sensors, triggers and actions. Triggers use sensor event data to conditionally launch actions. One novel aspect of MyExperience is that its behavior and user interface are specified via XML and a lightweight scripting language similar to the HTML/JavaScript paradigm on the web [9].

3.2 Overview of Our Mobile Activity Logger

As part of this work, we implemented 27 new hardware and software sensors and we used 11 existing sensors from the MyExperience project. Figure 2 gives an overview of available hardware and software sensors. Note that in our work a "sensor" more precisely is a piece of code that either connects to an actual hardware sensor on the mobile device, or reacts to software events or user input.

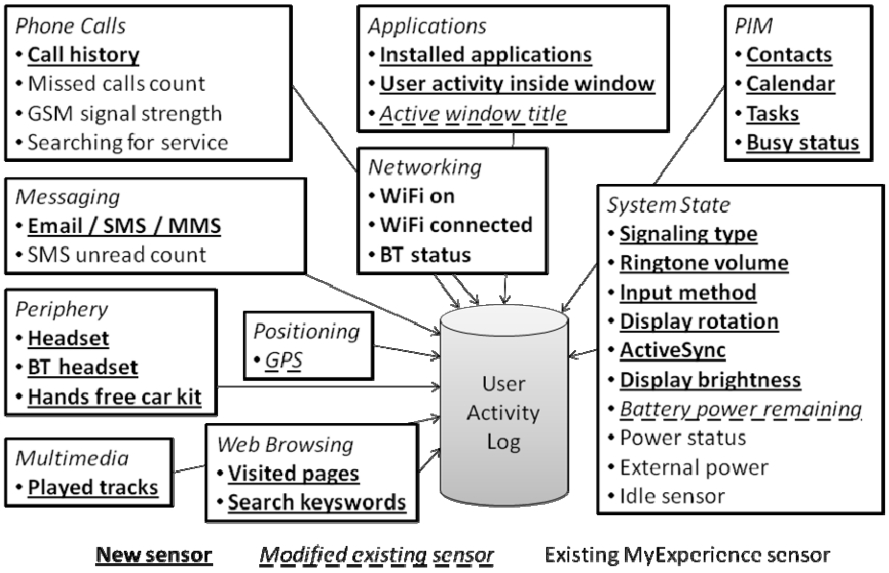


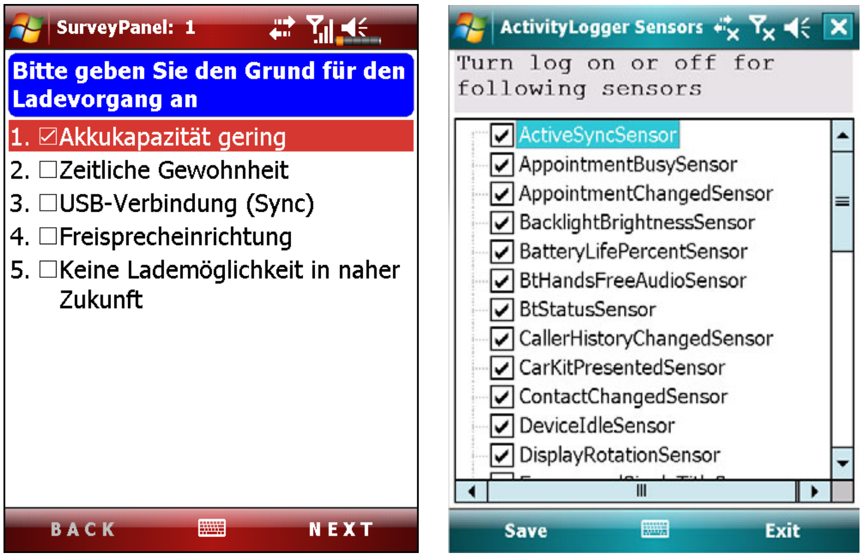
Fig. 2. Available sensors

MyExperience allows for configuring sensors via an XML file [9]. Note that the configuration not only controls which sensors to use for data recording, but MyExperience sensors also trigger actions such as starting an explicit user dialogue (Fig. 3, left). Since it is not viable to ask the end user to modify XML files on the mobile device, we have implemented an easy-to-use interface to activate and deactivate individual sensors (Fig. 3, right). Users may want to disable sensors for privacy reasons, and also to reduce power consumption or CPU load on the mobile device. The activity logger itself can be started and shut down manually by the user if necessary.

The implementation of sensors for implicit data acquisition can be summarized into the following categories:

- Application information such as visited web sites is usually stored in log files and local databases.
- Some sensors such as battery power status can be queried by using "SystemState" members of the .NET Compact Framework.
- Information about the location of local log files and some system information, such as display orientation and brightness, is available via the registry of the mobile device.

We will explain the available hardware and software sensors and issues concerning their implementation in more detail in the following subsections.


SurveyPanel: 1

Bitte geben Sie den Grund für den Ladevorgang an

1. ☑Akkukapazität gering
2. ☐Zeitliche Gewohnheit
3. ☐USB-Verbindung (Sync)
4. ☐Freisprecheinrichtung
5. ☐Keine Lademöglichkeit in naher Zukunft

ActivityLogger Sensors

Turn log on or off for following sensors

- ☑ ActiveSyncSensor
- ☑ AppointmentBusySensor
- ☑ AppointmentChangedSensor
- ☑ BacklightBrightnessSensor
- ☑ BatteryLifePercentSensor
- ☑ BtHandsFreeAudioSensor
- ☑ BtStatusSensor
- ☑ CallerHistoryChangedSensor
- ☑ CarKitPresentedSensor
- ☑ ContactChangedSensor
- ☑ DeviceIdleSensor
- ☑ DisplayRotationSensor

Fig. 3. Requesting explicit user feedback (left), and selecting sensors (right)

3.3 Sensors

3.3.1 Phone Calls

Making phone calls is one of the most important features of mobile devices. The fundamental parameters of a phone call are the phone numbers, the direction of the connection (outgoing, incoming, calls not accepted), the timestamp and the duration of the call. Furthermore, if the number of the other party can be found in the user's address book, additional information like name and group membership (e.g. family, friend) can be determined. This information could be used to suggest a callee's phone number, for example, when a user accesses the phone function of her device at a certain day and time of the week.

The .NET Compact Framework offers a possibility to setup an event handler for incoming calls. However, a handle to log outgoing calls is not provided. Yet logging outgoing calls is important for mobile user modeling, because they are the direct result of a user action. Therefore, we implemented a sensor to log the stated information about all phone calls. This sensor uses a list of all calls the Windows Mobile operating system keeps in a file in the Embedded Database (EDB) format. Our framework uses this list to retrieve the call parameters, and conducts a reverse search in the user's address book to determine more information about the other party of a call if available.

We integrated sensors to count missed calls, for GSM signal strength and searching for service from the MyExperience project without modification.

3.3.2 Messaging and Personal Information Management (PIM)

Windows Mobile provides the Microsoft Office Outlook Mobile Tool for managing Emails and SMS messages. The program splits information about messages by different accounts. Access to the internal Outlook database is possible with a wrapper library "MAPIdotnet" in the "Messaging API" of the .NET Compact Framework. We used this API to retrieve information about incoming and outgoing messages. Our corresponding software sensor records one log entry for every Email/SMS/MMS message. Similar to phone calls, we store additional information of the sender or recipient of a message if the person can be found in the user's address book.

Outlook Mobile is also the default tool for personal information management (PIM) on a Windows Mobile device. Appointments (calendar), contact data or tasks (ToDo lists) are interesting categories of data for user modeling as well. We have implemented sensors to log changes a user makes in her PIM data. Basically, there are two options. The first one is to monitor the data in the local Outlook database "pim.vol". We can recognize changed entries by comparing the Outlook IDs before and after the usage of Outlook. We have implemented separate but similar sensors for calendar, contacts and tasks. These sensors are triggered when the system recognizes that Outlook is called up or shut down. The second option to log Outlook data is based on an event handler. In this case, the system is immediately modified when the user adds, modifies or deletes data in Outlook. Our sensor then generates a log entry that includes the ID of the corresponding data item.

3.3.3 Web Browsing

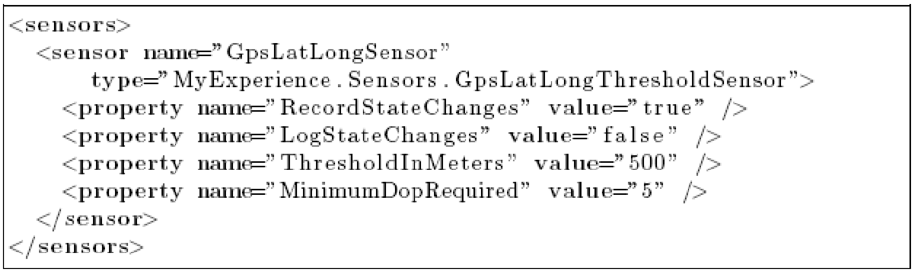
Analyzing the web browsing activities on the mobile device is an important part of mobile user modeling. We have created a MyExperience action to capture the usage of the Pocket Internet Explorer (PIE). It is not possible to directly query visited web sites in the .NET Compact Framework. However, the PIE manages information about visited web sites, cookies and temporary Internet files (cache) in three local files in different folders. The location of these files can be determined using the Windows Mobile registry. Access to these files is not permitted by the system if the PIE is running. Therefore, our sensor checks access to these files on start-up, and synchronizes the data with the activity log. The system keeps a timestamp of every accessed URL and visited web sites can hence be added to the activity log later on.

It is not only interesting for user modeling that a user has visited a web site, but also which keywords she has used for web searching. This information can be determined by analyzing the URL of web searches. For example, a query with Google leads to an URL similar to "http://www.google.com/search?q=activity+logging" in the log file. URLs to other search engine are comparable. We analyze and store the search keywords of about 20 search engines including Google, Yahoo and Bing, and also the query strings when accessing Wikipedia. Figure 5 (below) includes an example snapshot of recorded web browsing information.

3.3.4 Positioning

Obviously, one of most important differences between mobile and non-mobile systems is that the current user location is important in a mobile environment. Therefore it is important to log the user position in a mobile user modeling framework. There are a lot of work going on with regard to positioning systems, including approaches based on cell ID and WLAN access points. Since more and more mobile devices are equipped with a Global Position System (GPS) sensor, we decided to integrate GPS positioning in our framework.

The MyExperience project already includes a “GpsLatLongSensor” to trace GPS position. This sensor records GPS coordinates every one second. However, this leads to a lot of redundant GPS coordinates being stored in the user activity log, which is not relevant for mobile user modeling. Therefore, we have extended this sensor with an option to configure a threshold. The threshold triggers when the parameterized distance to the last recorded location in meters is surpassed. Figure 4 shows an example configuration for our GPS logging sensor.



```
<sensors>
  <sensor name="GpsLatLongSensor"
      type="MyExperience.Sensors.GpsLatLongThresholdSensor">
    <property name="RecordStateChanges" value="true" />
    <property name="LogStateChanges" value="false" />
    <property name="ThresholdInMeters" value="500" />
    <property name="MinimumDopRequired" value="5" />
  </sensor>
</sensors>
```

Fig. 4. Configuration of the “GpsLatLongThresholdSensor” sensor

First tests with this sensor revealed problems with weak GPS signals. Especially when activating the GPS sensor, or leaving a building with no signal, the first log entries sometimes deviated from the actual position by several kilometers on our test devices. In addition, the system sometimes recorded “(0, 0)” coordinates with no signal. These phenomena are not a problem when using GPS for navigation, for example, because the system quickly calibrates itself and then provides correct coordinates. However, we aimed at avoiding these false values in our user activity logs. Thus, we implemented a solution based on the “dilution of precision” (DOP) parameter of GPS sensors. This value is determined by the GPS sensor itself and specifies the additional multiplicative effect of GPS satellite geometry on GPS precision. The lower the value, the more accurate the measurement. We obtained good results – i.e. inaccurate log entries were eliminated – with a minimum DOP value of 5 in our tests.

3.3.5 Networking and Peripheral Devices

State-of-the-art mobile devices usually support several technologies for wireless connectivity, including GSM, Wireless-LAN/Wifi and Bluetooth. A mobile system could utilize information about networking usage to automatically activate and deactivate connections based on past user behavior. We give a detailed example as a case study in our evaluation in Section 4. We have implemented different sensors to log when the user has turned on WiFi access, when the system is actually connected to a WiFi access point, and the Bluetooth connection status. Furthermore, our framework provides sensors to monitor peripheral devices such as a headset or Bluetooth hands free kits often used in cars.

3.3.6 Application Usage and Media Player

A mobile user modeling framework should be able to derive a possible correlation between application usage and sensor data. MyExperience offers a sensor to retrieve the title of the active window and thus determine the active application. However, it is possible that some applications are missed because this sensor queries the system periodically to determine this value. Therefore, we have modified this sensor using an event handler. In addition, our framework offers a sensor to log installed applications.

Logging user action inside an application is difficult in the Windows Mobile operating system, because this information is generally available inside the active process only. Therefore it is not possible to record the text a user enters on the virtual keyboard in a text-processing program directly, for example. As an exception, the keystrokes on the hardware keys on a Windows Mobile device can be retrieved.

It is possible to build sensors for specific applications to be able to log more detailed information about application usage. As an example, we have implemented a sensor that logs the played tracks in the Windows Media Player. This information could be utilized later to provide context-aware media recommendations to the user.

3.3.7 System State

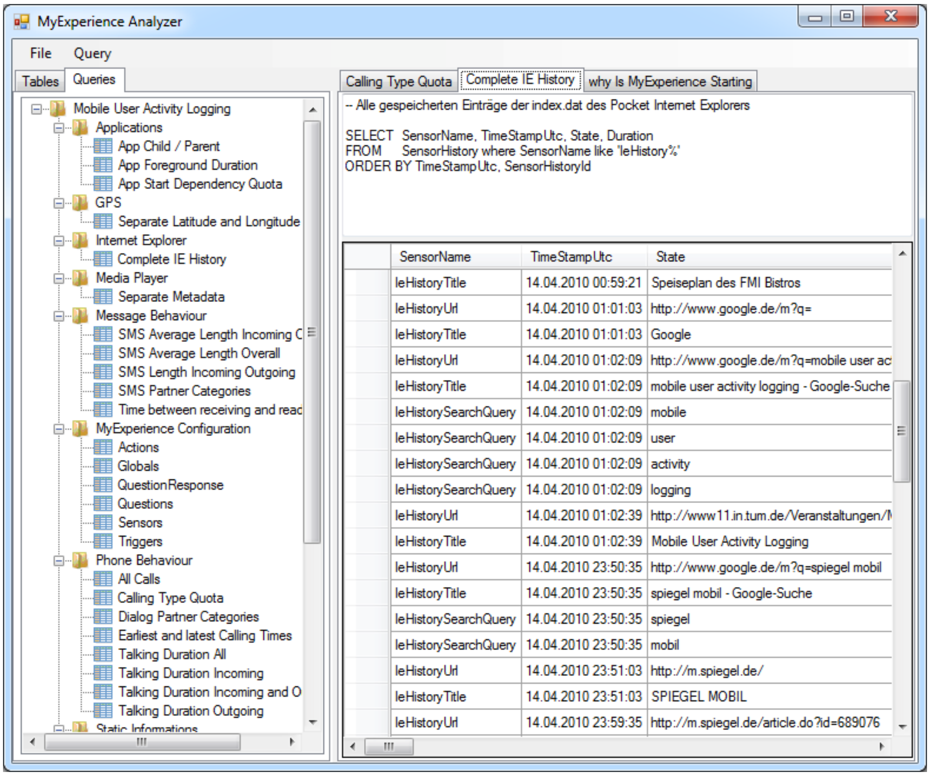
Finally, the last category of implemented sensors in our mobile user activity framework includes sensors that query the system state. Changes in the system state can be either triggered by user actions or an indirect result from usage of the device, for example battery power. Both are interesting for user modeling. Figure 2 lists the sensors we have implemented with regard to system state. An example is a sensor logging the input method a user selects. This information can be utilized to automatically select the appropriate input method based on previous user behavior. Windows Mobile devices with a touch screen usually offer a virtual keyboard and handwriting-recognition method such as Block Recognizer, Letter Recognizer or Transcriber. We implemented most of these system sensors by querying “SystemState” members in the “Microsoft.WindowsMobile.System” namespace of the .NET Compact Framework. The selected signaling type (vibration or ring) can be determined by querying a registry entry, in this case the variable “HKC\\ControlPanel\Sounds\RingTone0”.

3.4 MyExperience Analyzer Tool

The MyExperience framework stores all the information in a Microsoft SQL CE (Compact Edition) database on the mobile device [8]. The most important database table for our purposes is “SensorHistory” which stores the implicitly recorded data from the explained sensors. The MyExperience framework includes an “Analyzer” tool to manage and query the SQL CE database. We have extended this program with options to save and categorize queries. Queries are kept in an XML file, so it is possible to use them outside of the Analyzer tool.

Figure 5 depicts a screenshot of the extended Analyzer tool. On the left side, you can see the query library. This list corresponds to the implemented sensors in the categories explained above. On the right side of the window, an example query is shown. On top is the SQL CE query necessary to retrieve the Pocket Internet Explorer log entries.

Joining information from different sensors is possible but lead to rather complex SQL CE queries if done directly in the database. The Analyzer tool is intended to roughly check the collected data, not to interpret the gathered log data. For analysis and interpretation, the data can be exported from the database and further processed in data mining or other tools. For example, it is straightforward to analyze where the user has performed certain actions. This is also included in the following evaluation section.


MyExperience Analyzer
File
Query
Tables
Queries
Mobile User Activity Logging
Applications
App Child / Parent
App Foreground Duration
App Start Dependency Quota
GPS
Separate Latitude and Longitude
Internet Explorer
Complete IE History
Media Player
Separate Metadata
Message Behaviour
SMS Average Length Incoming C
SMS Average Length Overall
SMS Length Incoming Outgoing
SMS Partner Categories
Time between receiving and reac
MyExperience Configuration
Actions
Globals
QuestionResponse
Questions
Sensors
Triggers
Phone Behaviour
All Calls
Calling Type Quota
Dialog Partner Categories
Earliest and latest Calling Times
Talking Duration All
Talking Duration Incoming
Talking Duration Incoming and O
Talking Duration Outgoing
Static Informations
Calling Type Quota
Complete IE History
why Is MyExperience Starting

```
-- Alle gespeicherten Einträge der index.dat des Pocket Internet Explorers

SELECT  SensorName, TimeStampUtc, State, Duration
FROM    SensorHistory where SensorName like 'IeHistory%'
ORDER BY TimeStampUtc, SensorHistoryId
```

SensorName	TimeStampUtc	State
IeHistoryTitle	14.04.2010 00:59:21	Speiseplan des FMI Bistros
IeHistoryUrl	14.04.2010 01:01:03	http://www.google.de/m?q=
IeHistoryTitle	14.04.2010 01:01:03	Google
IeHistoryUrl	14.04.2010 01:02:09	http://www.google.de/m?q=mobile user ac
IeHistoryTitle	14.04.2010 01:02:09	mobile user activity logging - Google-Suche
IeHistorySearchQuery	14.04.2010 01:02:09	mobile
IeHistorySearchQuery	14.04.2010 01:02:09	user
IeHistorySearchQuery	14.04.2010 01:02:09	activity
IeHistorySearchQuery	14.04.2010 01:02:09	logging
IeHistoryUrl	14.04.2010 01:02:39	http://www11.in.tum.de/Veranstaltungen/M
IeHistoryTitle	14.04.2010 01:02:39	Mobile User Activity Logging
IeHistoryUrl	14.04.2010 23:50:35	http://www.google.de/m?q=spiegel mobil
IeHistoryTitle	14.04.2010 23:50:35	spiegel mobil - Google-Suche
IeHistorySearchQuery	14.04.2010 23:50:35	spiegel
IeHistorySearchQuery	14.04.2010 23:50:35	mobil
IeHistoryUrl	14.04.2010 23:51:03	http://m.spiegel.de/
IeHistoryTitle	14.04.2010 23:51:03	SPIEGEL MOBIL
IeHistoryUrl	14.04.2010 23:59:35	http://m.spiegel.de/article.do?id=689076

Fig. 5. Analyzer screenshot

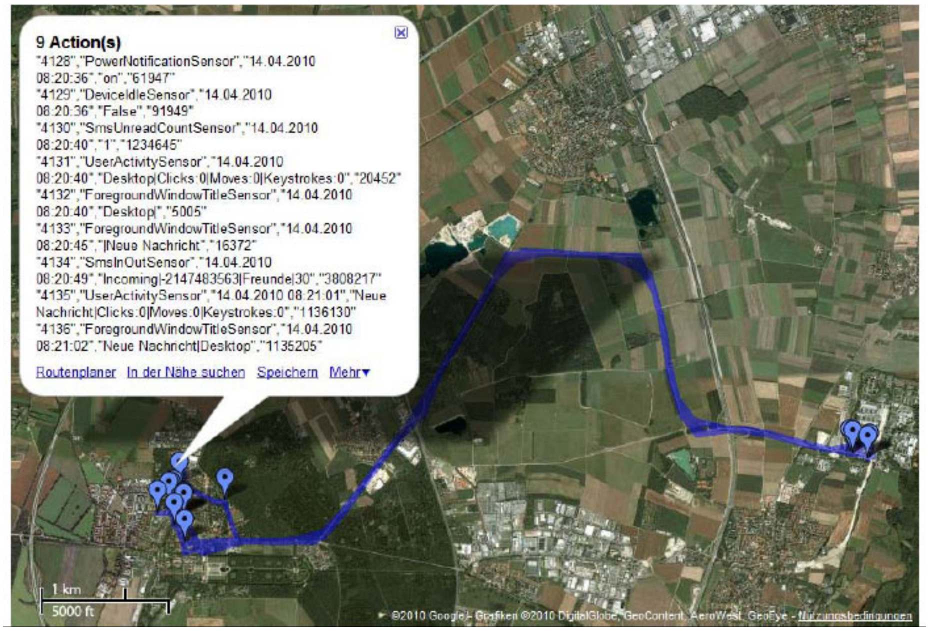
4 Evaluation of the Collection of User Data

In this section, we explain the evaluation of our approach. Note that we have focused on the collection of user data only so far. Therefore, our approach and the evaluation do not cover the whole user modeling process (see introduction, Section 1), just the first step.

4.1 Experiences

We tested our sensors during implementation to make sure they perform accurately. Afterwards, we conducted a test run of the system lasting several weeks and included all sensors. During this time, 6748 log entries were recorded. In addition, we looked at scenarios to find out whether the recorded data can lead to meaningful data for user modeling. Meaningful data means information about the user that is characteristic for a situation the user is in. We describe one of these scenarios as a case study in chapter 4.2. We did not include explicit user feedback (Fig. 3, left) in these tests, but this function could have been integrated easily.

Figure 6 depicts a visualization of parts of the logged data. For this visualization, the GPS position data was converted into the GPS Exchange Format (GPX) for Google Maps. The (blue) markers show locations where the user performed some activity on the mobile device. The figure depicts a typical scenario when a user travels from home to office during a workday. When looking at the data more closely, we were able to assess that the log files reproduced the user actions very well and in reasonable granularity for user modeling. Another example of the logged data is shown in the screenshot of the analyzer in Figure 5 (above). Thereby, the user was using her Pocket Internet Explorer to perform some web searches and the keywords of the searches were detected by the system.


9 Action(s)

```
"4128","PowerNotificationSensor","14.04.2010
08:20:36","on","61947"
"4129","DeviceIdleSensor","14.04.2010
08:20:36","False","91949"
"4130","SmsUnreadCountSensor","14.04.2010
08:20:40","1","1234645"
"4131","UserActivitySensor","14.04.2010
08:20:40","Desktop|Clicks:0|Moves:0|Keystrokes:0","20452"
"4132","ForegroundWindowTitleSensor","14.04.2010
08:20:40","Desktop|","5005"
"4133","ForegroundWindowTitleSensor","14.04.2010
08:20:45","|Neue Nachricht","16372"
"4134","SmsInOutSensor","14.04.2010
08:20:49","Incoming|-2147483563|Freunde|30","3808217"
"4135","UserActivitySensor","14.04.2010 08:21:01","Neue
Nachricht|Clicks:0|Moves:0|Keystrokes:0","1136130"
"4136","ForegroundWindowTitleSensor","14.04.2010
08:21:02","Neue Nachricht|Desktop","1135205"
```

Routenplaner
In der Nähe suchen
Speichern
Mehr▾
1 km
5000 ft
©2010 Google - Grafiken ©2010 DigitalGlobe, GeoContent, AeroWest, GeoEye - Nutzungsbedingungen

Fig. 6. Visualization of log data

Overall, our mobile logging framework performed well. There were a few program crashes in the prototype implementation but these occurred only very seldom. When the user was very active on her device and all sensors were enabled, the system performance degenerated somewhat. However, it is possible and assumed that not all sensors are active at all times. It is possible to deactivate sensors as explained above

(Chapter 3.2). Overall, the logging did not obstruct the user experience significantly. Thus, our system complied with one of our main requirements: the implicit, non-distracting observation of user actions.

Apart from that, we have to note that, for an ongoing recording of sensor data, an active system status is required. Windows Mobile PDAs are usually configured to be hibernated when the user is inactive for some time. When this occurs, our logger is also stalled of course. However, with an inactive system, no meaningful user actions can be recorded anyways. If the user just turns off the display of her device, the recording of sensors such as battery power or GPS position continues. The battery power is shortened to a couple of hours at most without charging when all sensors are activated, but again the power consumption can be reduced by deactivating costly sensors such as the GPS module. It seems reasonable to define profiles with different sensors active (e.g. "indoor" with a disabled GPS sensor, or "light" with only a small subset of sensors active). Users would only have to choose among predefined profiles, not all sensors. But this profiling feature has not been implemented yet.

4.2 Case Study: WLAN Activation Based on User Position

In this scenario, we had a closer look on whether it is possible to identify locations where a user usually activates the WiFi/WLAN connection on her mobile device. The overall goal is that the system would then be able to automatically turn on WiFi when the user enters such a region. Thus, a combination of the "GpsLatLongThresholdSensor" with the "WiFiConnectedSensor" is investigated. In this test, the user moved her mobile device in an area with two WLAN access points. The GPS logging was set to store one log entry every 10 meters. The recorded position data was combined with the WiFiConnectedSensor data based on the timestamps of log entries. Figure 7 shows the graphical interpretation of the data. Dark (red) dots mark GPS positions with no WLAN activated, while the light (green) markings denote positions where the user has turned on WLAN. The (manually) highlighted areas indicate these geographic regions where the user usually activated her WLAN connection.

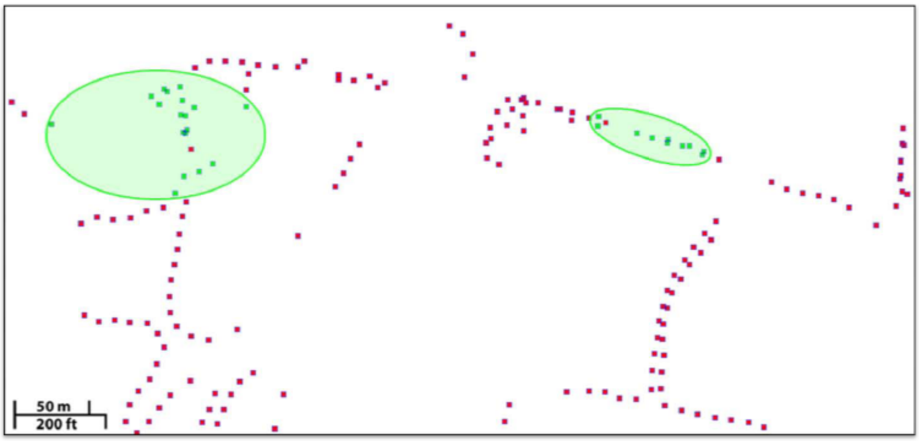


Fig. 7. WLAN activation based on position

The recorded data corresponds very well with the actual WLAN access areas. We noticed that some of the points were slightly off when the user was moving fast. This behavior is due to slight delays when the system is observing and logging the deactivation of WLAN access. Overall, the recorded data seemed to be very useful for our purpose. Note again that the goal of this scenario was to evaluate whether the collected data can lead to meaningful results for user modeling. The scenario showed that a combination of sensors can be used to implement an adaptive function to automatically activate the WiFi/WLAN connection on the mobile device based on location. Our tests showed that our mobile user activity logger produced data in appropriate granularity for user modeling.

5 Mining the Logged Data Sets

While the focus of this project so far lies on the acquisition of data, the ultimate goal is to derive information about the user from analyzing the gathered collection of raw data. For this purpose, we can avail ourselves of some established methods from the field of data mining. We are currently working on examining which data mining methods fits our data best. We have also gained some promising early results. This section serves as an overview of possible applications for further work.

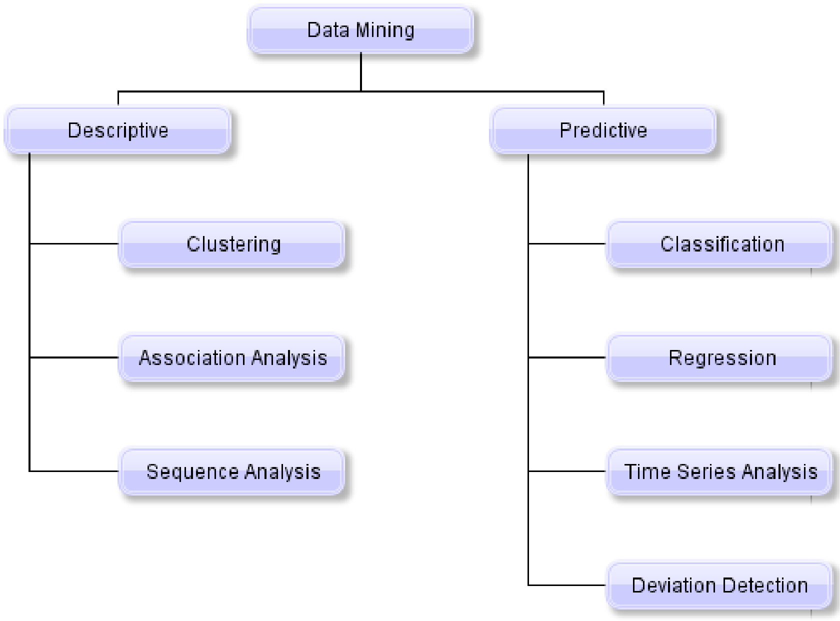


Fig. 8. Data mining tasks

As shown in Figure 8, two branches of common tasks in data mining can be distinguished: descriptive methods try to identify patterns and relations in the data giving statements on their properties, whereas predictive methods make new assumptions based on known information.

5.1 Clustering

Clustering is the task of partitioning a set of (multidimensional) data points into groups of high intra-group similarity according to a chosen measure. Each cluster can then be labeled manually and treated accordingly. As a practical example, the sensor readings of a device with steadily activated GPS will most likely create clusters in the data set that represent geographical areas. This is due to the fact that a user is assumed not to be moving constantly. When there is little or no movement the data points will concentrate in a small region. Accordingly, each cluster marks a place at which the user remained for a significant amount of time. Unfortunately, most of the common clustering algorithms like for example k-means clustering require the number of clusters as predetermined parameter. Finding the optimal value iteratively by running the process multiple times proves unfeasible on a mobile device.

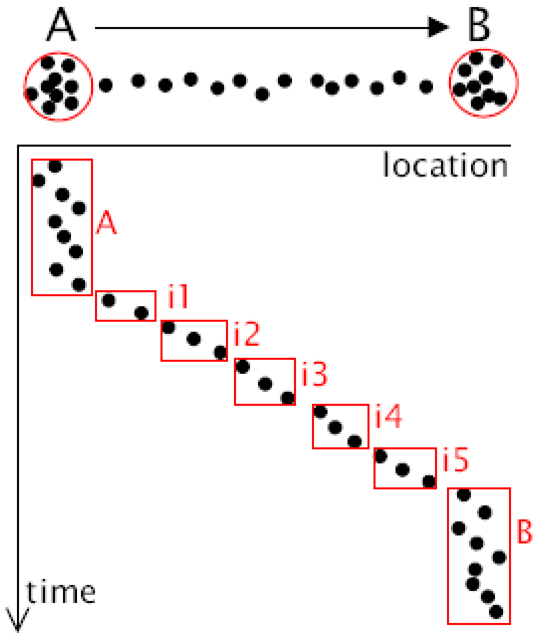


Fig. 9. Time-based clustering [10]

To overcome this shortcoming, [10] propose a time-based approach which clusters the stream of incoming location coordinates along the time axis and drops the smaller clusters that represent places with only a short length of stay. The algorithm compares each incoming coordinate with previous coordinates in the current cluster; if the new coordinate is moving away from the cluster beyond a certain threshold, a new cluster is formed (cf. Figure 9). Along the user's way from location A to location B, several intermediate clusters of smaller size are built If a cluster's time duration is below a second threshold that cluster is considered insignificant and, therefore, dropped. That way, only places at which the user spends a certain amount of time are detected.

This incremental and computationally simple approach allows for the extraction of location at run-time, even on a mobile device. Now, as raw GPS coordinates provide no semantics, we can map them to real places using reverse geo-coding. It is also possible to establish a hierarchy of locations by extracting the common content of multiple mappings, e.g. the name of a street. This procedure can be repeated successively, each time yielding an ontological representation of higher abstraction for a geographical region.

We have implemented and tested the time-based clustering approach by Kang et. al. [9] for Microsoft Windows Mobile PDAs. One significant advantage of perfoming the analysis locally on the mobile device is privacy. The user data never leaves the PDA and the user has full control over the whole data collection and mining process. The original goal in this subproject was to identify relevant user locations for semantic personal information management [11]. In this application scenario, users can utilize a personal ontology to manage resources. To do so, the system proposed relevant location in the ontology. But the generated relevant user locations can also be used for other application scenario.

The clustering algorithm produces a list of GPS coordinates – longitude and latitude, e.g. "(48.1300995333333, 11.5934058166667)". Since we do not need coordinates but named locations in this scenario, we need to perform reverse geo-coding. These services translate GPS coordinates in actual place names. We selected and used the "Google Reverse Geocoder" which is part of the Google Maps API family. For performance reasons, we implemented a client-side cache to store earlier geo-coding results [11]. The GPS example above translates into "Rosenheimer Straße 6-64, 81667 Munich, Germany", for example. Hence, we gained a flat list of possibly relevant addresses. In the last interpretation step, we need to build a hierarchy of locations. To do so, our application extracts place names that appear in more than one relevant address.

We evaluated this approach in a small user study with seven test users and HTC P3600 PDA phones equipped with GPS [11]. The study lasted about two weeks and took place in the German city Munich. We could not use the data of one user because of a malfunction of the GPS sensor on the test device. After the interpretation of the data as explained above, the system presented the user with the determined locations and we asked them to judge the relevance of this location and whether they would include the entry in their personal ontology or not. The relevance was ranked on a scale 1-5 with 5 meaning that a location is very relevant for a user. In addition, the users were asked to record the actual addresses they were visiting during the test period. This allows for recognizing locations missed by the algorithm [11].

We analyzed the results according to the metrics precision, recall and also "usefulness". Precision is the measure of exactness and is defined by the number of relevant locations divided by the total number of locations our approach proposed. Recall is the measure of completeness and indicates the number of relevant locations found by our algorithm divided by the number of addresses a user visited in our scenario. A relevant location is a location the user would include in her personal ontology. This means, a determined location may not be wrong, but the user just would not deem it important, for whatever reason. In addition, we looked at the usefulness of a proposed location. This parameter utilizes the relevance score the users indicated for each address, and weighs recall according this relevance. Usefulness is thus based on an explicit weight, the users assigned to locations. Table 1 shows the results from our small experiment [11].

Table 1. Experimental results

	Precision	**Recall**	**Usefulness**
User 1	0,66	0,66	0,66
User 2	0,66	0,33	0,16
User 3	1,00	0,66	0,50
User 4	1,00	0,80	0,60
User 5	0,66	0,66	0,66
User 6	0,75	1,00	1,00
Total:	**0,788**	**0,685**	**0,596**

Our approach based on the time-based clustering approach by Kang et.al. [10] detected almost 80% of the relevant locations for a user. As a baseline, we compared the results with the offline clustering algorithm DBSCAN. Applying DBSCAN results in comparable outcomes for recall and usefulness, but a significantly lower value for precision: 0,399 in comparison to 0,788.

5.2 Mining Association Rules

Another very interesting task for our purposes is the discovery of so called association rules. These rules describe co-occurrence relationships among variables as an implication of the form $X \rightarrow Y$. The search for associations is based on search for frequent patterns. Imagine we have a computerized model that maps GPS data to locations. Now, if we detect frequent coincidence of an appointment, GPS sensor readings that correspond to the location associated with that appointment, and evidence of a muted phone then we might establish the following rule: {appointment, corresponding location} $\rightarrow$ mute phone. Another example is a music recommender on the basis of associating the location of the user with genre of music listened, e.g. recommending romantic music when the user is near a beach.

Closely related to both the abovementioned fields is the classification of data points. While clustering tries to partition the data set into groups, classification also attempts to find a decision function for future assignment of new data points to one of them. Association rules can serve as such a function if the rules predict a single target class in form of a definite value for a certain variable. For this purpose, for each rule the system checks whether all antecedents are existent at the given time to deduce the validity of the consequent. In that special case, association rules resemble a decision tree approach to classification. Applied to our case of mobile sensor data, we could use our set of readings to train a decision tree in order to solve the binary classification problem whether a given combination of sensor readings entails the activation of Wifi.

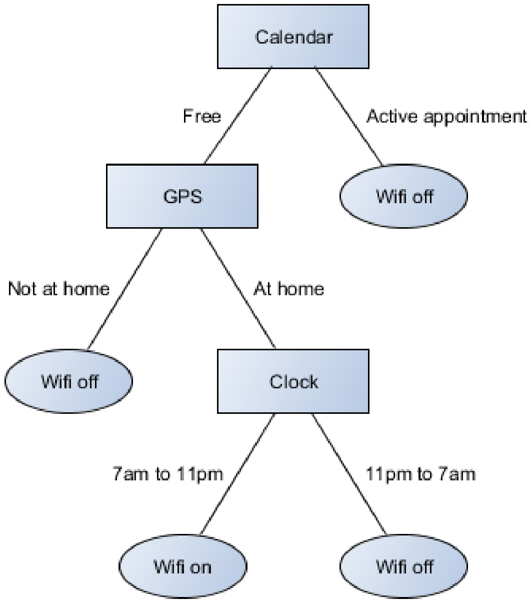


Fig. 10. Decision tree using sensor evidences for the decision of Wifi activation

Figure 10 depicts a decision tree created for this task based on a training set (cf. Table 2) comprising three sensors: the calendar, the GPS sensor, and the system clock. Only if these sensors indicate that the user is at home during the day with no active appointments the Wifi chip is activated. Note the synthetic nature of the thresholds for the purpose of clarification.

Table 2. Data set for training of the decision tree

Appointments	Location	Time	Wifi
none	Jackie's	11:43	off
none	Home	18:21	on
none	Home	23:33	off
Cinema	Cinema	11:51	off
none	Home	17:33	on
Dinner Peter	Home	14:12	off

Another useful task is the detection of deviation from identified patterns. Imagine a user who mutes his or her phone every evening. When getting up in the morning, he or she turns the sound back on. This daily routine marks a pattern in the according sensor data collection. Now, if one morning the ringer is not reactivated an outlier can be detected by the system and the question arises whether the user should be notified so he or she doesn't miss any incoming calls.

6 Conclusion and Outlook

The goal of this work is a unified approach for collecting data about user actions on mobile devices in granularity appropriate for user modeling. To realize this first step of the user modeling process, we have designed and implemented a framework for mobile user activity logging on Windows Mobile PDAs based on the MyExperience project. We have extended this system with hardware and software sensors to monitor phone calls, messaging, peripheral devices, media players, GPS sensors, networking, personal information management, web browsing, system behavior and application usage. Our evaluation showed that it is possible to detect when, at which location and how a user uses an application or accesses certain information, for example.

Note that collecting data about user actions is more complicated on a mobile device than a desktop setting. This is due to restrictions in the available programming interfaces of the mobile platforms, in our case the Windows Mobile operating system, respective the .NET Compact Framework. We have explained some of the details of implementing the sensors in Section 3. The granularity or level of detail of the data collection is obviously dependent on the purpose of a subsequent user modeling task. We have aimed at selecting and designing sensors that lead to information, which seems beneficial for learning user behavior in general. For example, our web browsing sensor records search keywords, but not single keystrokes a user may perform to fill in a web form. The framework can be used to implement new or modified sensors to fit special data collection purposes. In addition, properties of some sensors can be configured to adapt the data collection in more detail.

Future work includes integrating additional sensors. State-of-the-art mobile devices are more and more equipped with sophisticated sensors such as gravitation sensors or cameras that could be utilized for eye tracking. It is easy to integrate additional sensors in the MyExperience project and our framework. Portability and interoperability are also important issues. So far, our framework is tailored for the Microsoft Windows Mobile framework but similar tools can be implemented on other platforms like iPhone and Android. We are also investigating standards for interoperability of data collected on different platforms or logging frameworks. Existing relevant initiatives include the Attention Profiling Markup Language (APML) [12] and the Contextualized Attention Metadata framework (CAMf) [13].

One of the most important next steps of our work is also to investigate the analysis of the collected data using data mining and machine learning methods, and hence studying the second step of the user modeling process (see Section 1). We have outlined some preliminary ideas in this regard in Section 5 of this paper. One of our scenarios we have worked on is to identify relevant user locations for semantic personal information management on mobile devices [11]. We are currently working on analyzing various methods to mine the logged data sets and derive information about user behavior in more detail. It is also important to collect more substantial data sets in this regard.

Finally, our work focuses on observing one user so far. It may also be useful to take other users' logs into account and thus performing a "social" mobile user activity logging. The goal could be to identify situations where similar behaving users have performed certain actions, and personalize the mobile experience for the active user accordingly. In addition, our system also collects data about social interactions that can be utilized for analysis of social behavior. A software sensor could connect to social networking sites and log corresponding user activities.

References

1. Brusilovsky, P., Maybury, M.T.: From Adaptive Hypermedia to the Adaptive Web. Communications of the ACM 45(5), 30–33 (2002)
2. Subramanya, S.R., Yi, B.K.: Enhancing the User Experience in Mobile Phones. IEEE Computer 40(12), 114–117 (2007)
3. Froehlich, J., Chen, M., Consolvo, S., Harrison, B., Landay, J.: MyExperience: A System for In situ Tracing and Capturing of User Feedback on Mobile Phones. In: Proc. of MobiSys Conf., San Juan, Puerto Rico (2007)
4. Chernov, S., Demartini, G., Herder, E., Kopycki, M., Nejdl, W.: Evaluating Personal Information Management Using an Activity Logs Enriched Desktop Dataset. In: Proc. of 3rd Personal Information Management Workshop (PIM 2008), CHI Conf., Florence, Italy (2008)
5. Zheng, Y., Zhang, L., Xie, X., Ma, W.: Mining Interesting Locations and Travel Sequences From GPS Trajectories. In: Proc. of International Conference on World Wide Web (WWW 2009), Madrid, Spain, pp. 791–800. ACM Press, New York (2009)
6. Choudhury, T., et al.: The Mobile Sensing Platform: An Embedded Activity Recognition System. IEEE Pervasive Computing 7(2), 32–41 (2008)
7. Jeong, J., Won, J., Bae, C.: User Activity Recognition and Logging in Distributed Intelligent Gadgets. In: IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, Seoul, Korea (2008)
8. Kanjo, E., Bacon, J., Roberts, D., Landshoff, P.: MobSens: Making Smart Phones Smarter. IEEE Pervasive Computing 8(4), 50–57 (2009)
9. MyExperience project, `http://myexperience.sourceforge.net/` (accessed, June 2010)
10. Kang, J.H., Welbourne, W., Stewart, B., Borriello, G.: Extracting Places From Traces of Locations. ACM SIGMOBILE Mobile Computing and Communications Review 9(3), 58–68 (2005)
11. Woerndl, W., Schulze, F., Yordanova, V.: Modeling and Learning Relevant Locations for a Mobile Semantic Desktop Application. Journal of Multimedia Processing and Technologies (JMPT) 1(1) (2010)
12. APML, `http://apml.areyoupayingattention.com/` (accessed, May 2011)
13. CAMf, `http://www.ariadne-eu.org/index.php?option=com_content&task=view&id=39&Itemid=55` (accessed, May 2011)

Intentional Modeling of Social Media Design Knowledge for Government-Citizen Communication*

Andrew Hilts and Eric Yu

Faculty of Information
University of Toronto
140 St. George Street, Toronto, ON
{andrew.hilts,eric.yu}@utoronto.ca

Abstract. Social media can be employed as powerful tools for enabling broad participation in public policy making. However, variations in the design of a social media technology system can lead to different levels or kinds of engagement, including low participation or polarized interchanges. An effective means toward learning of and analyzing the complex motivations, expectations, and actions among various actors in political communication can help designers create satisfactory social media systems.

This paper uses the *i** modeling framework to analyze the impact that alternative configurations of a social media technology can have on the goals and relationships of the actors involved. In doing so, we demonstrate and provide preliminary validation for a research-informed model creation and analysis approach to assessing competing design alternatives in an online climate change debate community.

1 Introduction

Town hall meetings, radio call-in shows and citizen surveys have been traditionally used by politicians to learn of the issues facing their constituents. The goal of this process is often to aid government policy conceptualization [3,22,29] or policy feedback [9]. Increasingly, politicians are using social media as channels to support this citizen opinion elicitation. For example, YouTube has been employed by a number of politicians who answer most of the popular questions posed to them by their online audiences. In this regard, it is the collective opinion of social media users that help to set a political agenda [5].

Participatory policy making [33] can be supported by social media when a community collectively evaluates and assesses important issues. In many social media discussion environments, users assign a positive or negative valuation (a "vote") on each other's posted comments. This arrangement results in a 'collaborative filter'; the highest-valued comments are shown prominently, while

* This is an extended version of a paper [19] presented at the First International Workshop on Modeling Social Media, held in 2010.

content of low value is hidden from view[1]. Ideally the highest-valued comments would represent the rationally-determined important issues facing the community. However, 'online deliberation' scholars and policy makers are finding that variations in the arrangement and organization of the user-generated content result in changes to user contribution and 'democratic' debate [16,17]. These variations thus help to shape the system's underlying social values [22].

Design choices tacitly embedded in a technology help to constrain and facilitate user behaviour, participant goal achievement and interdependent relationships [34]. However, online deliberation-focused scholarly findings that could help inform system design activities are not being adequately synthesized, which limits the development of a reciprocal academic and practitioner community that builds on one another's work [8]. In pursuit of such reciprocity, we propose that by carefully referencing existing academic literature at an early stage in the design process, a designer may incorporate past evidence in an analysis of the impact of various system configurations on stakeholder goals and potentially avoid replicating some of the failures of past work.

Models support this kind of reasoning activities by abstracting domain concepts into a representation structured to aid in answering analysis questions. For example, as we have been discussing, a designer can model how different configurations may contribute to the success or failure of a system. This can be demonstrated in a model of the alternatives available to the designer of a collaborative filter situated in a government-citizen communication system. Such a system can benefit from precise, model-supported analysis of its complex socio-technical domain.

In this chapter, we describe a research-informed early requirements modeling approach meant to address the above issues. We first provide a concrete example of a context in which such an approach would be helpful, followed by a demonstration of the method. We conclude with a brief discussion of our ongoing work in further systematizing a framework for social media design knowledge codification and recontextualization.

2 Designs, Goals and Politics

A mark of a well-designed system is that it supports the satisfaction of stakeholder goals better than feasible alternative designs. By conceiving of the system in terms of goals to be achieved, rather than solutions to implement, designers can effectively consider the granular impacts of various design choices [6]. The utility of this approach increases alongside the complexity of the setting; a rigorous methodology can aid in understanding the multifaceted relationships of interdependent stakeholders with competing goals.

As a simple example, the designer of a collaborative political comment filter can benefit from analyzing the goals of the client politician supporting the project as well as those of the citizen users whose contributions are essential to

[1] See the comment sections of www.youtube.com, www.reddit.com, www.digg.com for popular examples.

the system. A basic functional requirement could be that the system organize the user-created content in such a way that the most important comments are the highest-ranked. However, there are various ways of interpreting what is meant by the "most important comments", and correspondingly, different possible configurations of the collaborative filter that would help to fulfill the requirement.

This scenario is a concise example used to demonstrate our approach to modeling design alternatives in a socio-technical setting. In practice, it is likely that a designer would consider many diverse methods for eliciting and rating user-generated content. Collaborative filtering, our focal design concept, is but one of these potential methods.

2.1 Goals and Design Alternatives

In our collaborative filter example, the criteria that define an 'important' comment depend on the goals of the stakeholders, such as the client political institution. Based on the literature, politicians often desire to advertise themselves [3] and advantageously frame a publicly discussed issue [21] in order to manage citizen expectations on what is plausible [15]. Therefore, depending on the issue, the politician may desire that shared opinions be framed either as a consensus or as an open debate. Consequently, an 'important comment' would be defined either as 'popular' or 'controversial'.

Based on a recognition of the above goals, a system designer can choose to configure the collaborative filter either to highlight popular and agreeing opinions or conversely, diverse and conflicting views. The former design alternative, hereafter referred to as the "complementary filter" would compare user comments, voting activity, and profile information with those of other users in order to display familiar content and ideas to the user. This would likely encourage users to consolidate and clarify existing positions [4]. The latter alternative, the "contestatory filter" would compare the same items, but display unfamiliar and potentially challenging content, likely inciting debate [11]. These alternative designs of a collaborative filter would have variable impacts on the goals of multiple stakeholders, as described below in Section 3.3.

Concurrently, important comments must fuel the sustained activity of the community of citizen users. Otherwise, there would not be enough activity on which the collaborative filter bases its processing; the filter thus would not be able to present a comprehensive overview of citizen opinion to the politician stakeholder. As such, a designer's definition of an important comment must also consider the concept's relationship to citizen goals.

The literature suggests a wide range of motivating factors that encourage or discourage a citizen's opinion expression in a web-based political setting. Primarily, a vocal citizen is interested in and feels connected to the topic issue [22,28,31]. These motivations may be encouraged by the ability to personalize an information system and filter information based on interest [23,29]. The degree to which the citizen has faith in and a connection with government [22,23], a sense of citizen identity [28], and the sense that participation generates meaningful outcomes [23,36,25] are all shown to encourage citizen involvement.

Based on a synthesis of scholarly literature related to the domain and actual stakeholder interviews, the designer may conceive of how alternative design choices based on one stakeholder's goals affect those of other, interdependent stakeholders. In our particular case of the collaborative filter, the designer would consider how the employment of either a complementary or contestatory filtering mechanism might affect the participation of citizens. High-level system requirements could then be derived from this analysis.

However, as interrelationships among the goals, actors and design features under consideration become more concretely understood, the designer's ability to simultaneously consider all potential factors may be reduced. Complex contexts require analytic approaches designed to deal with this complexity. As mentioned above, prescriptive modeling of software processes, domains and organizational dependencies have been used effectively in software engineering and information systems design disciplines as a means to help deal with such problems.

2.2 Modeling the 'Why' Questions

Goal- and agent-oriented requirements engineering modeling techniques have been recognized as effective means of eliciting organizational-level requirements based on stakeholder needs [6,7]. Models of this sort generally depict a tree-like hierarchy of goals. High-level 'root' goals are abstract, generalized concepts, while lower-tier, 'leaf' elements are more specialized and concrete. These goals may be conceptualized as functional software requirements, software qualities, or stakeholder objectives[2].

By employing such a technique, the designer may decide on specific design feature configurations based on their efficacy in fulfilling the stakeholder goals in question. Proponents of this approach argue that it facilitates analysis centered on the design *problem*, rather than on a particular *solution* [37]. Alternative means towards the accomplishment of the same goal can be simultaneously considered as competing solutions with different benefits and trade-offs associated with each.

This approach enables knowledge from the literature to be concisely expressed in easily understandable 'means-ends' relationships. This focus on relational codification is quite similar to other design knowledge reuse approaches from the architecture and engineering fields. For example, architectural knowledge has been stored as a series of 'precedents', each made up of a relationships among design issues and the concept and form of the designed solution [30]. Other approaches include a method from aerospace engineering that divides design knowledge into issues, the process of design, the product, and the function of the artifact [2]. An approach from the requirements engineering discipline structures knowledge of agile software development 'method fragments', which are related to 'objectives' those fragments may satisfy and 'requisites' which are required for the fragment to function properly [12].

These approaches all assign a degree of importance to the goals, objectives, or issues that a specific design feature can address, though less explicitly so

[2] Among other definitions.

than the means-ends relationships codified in goal models[3]. However, they lack a formal means of representing relationships between interdependent actors, which are arguably highly important factors when designing *social* media systems. As such, a more useful representational scheme would include a means of modeling such relations.

A prominent goal- and agent-oriented modeling framework is *i** [38], which focuses on systems of intentional strategic actors. Due to its focus on relationships, we believe this approach is well suited for the modeling of socially-situated information systems design knowledge. Indeed, a modified version of the framework has previously been used in the modeling of an e-government service [10]. We have extensive experience using this modeling approach, and now turn to a brief introduction of its major concepts.

2.3 The *i** Modeling Approach

The central modeling construct in *i** is the *actor*, an entity whose autonomous behaviour is based on reasons and motivations. *i** models conceive of the social world as a network of interdependent relationships; an actor may *depend* on another to fulfill a *goal*, furnish a *resource*, or carry out a *task*. If an actor depends on another and that dependency is not met, the actors own internal goals may fail.

In this manner, *i** can depict the relationship between individual actor intentionality and the broader social setting. Furthermore, an intentional ontology allows for the analysis of 'why' an actor may prefer one possible design alternative over another.

This contrasts with KAOS, another prominent goal modeling framework, where actor interdependency is not explicitly related to the represented goals [35]; the model is somewhat decontextualized. This is not the case with *i**, and thus this technique provides suitable support for a designer's contextually situated, goal-based analysis of design choices. Indeed, an empirical assessment of *i** has found that the modeling framework provides valuable expressive power for representing and assessing the social relations of interdependent stakeholders as well as how certain activities can impact their goals [32].

There are several important conceptual elements which make up the *i** meta model. The above-described *actor* may desire to fulfill various *soft goals* – elements that depict quality characteristics that cannot be objectively satisfied or denied, but require a more qualitative assessment of their acceptability. *Goals* (or *hard goals*), are more concrete, clearly satisfiable objectives. *Tasks* are means to achieving a goal, or contributing to a Soft Goal. *Resources* are often required for tasks to be completed.

Importantly, tasks may be used to model competing design feature alternatives. For example, two tasks could both be modeled as means towards an actor's goal. However, the tasks might also have varying *contribution links* to other goals or soft goals. In this regard, *i** can clearly indicate how different system

[3] With the exception of [12], which is goal-oriented.

configurations might variably affect stakeholder goals. An *i**-specific evaluation methodology [20] provides a clear process for analyzing these contributions and supporting the selection of satisficing design feature configurations. We employ *i** in the following sections as a framework for codifying and analyzing the effects of design alternatives in a government-citizen communication context.

3 A Demonstration of Research-Informed Domain Model Creation

The following modeling demonstration exemplifies the potential of goal- and agent-oriented modeling methodologies in aiding early stage requirements analysis of social media for online deliberation. We undertook a small exploratory case study, and proposed a sample technology – the above-described collaborative filter – to be introduced in the design of an climate change online debate community.[4]

Many *i** and related modeling approaches employ knowledge extracted from the literature as a source for model construction [6,12,27]. We build on this work by explicitly focusing on a clear method for research codification, abstraction and synthesis as a means toward model construction. This approach should be seen as complementary to traditional requirements gathering activities in an application environment [13]. As such, recalling our earlier discussion of the context-dependent characterization of an "important comment", it is important that requirements gathering efforts unearth such constraints and consider how they will interact with less contextually situated relationships uncovered from the literature. Therefore, when they are employed in design projects, research-informed design knowledge models should be contextually tailored and synthesized with other requirements analysis efforts. Below, we demonstrate an approach to doing so.

3.1 Model Construction

In order to extract relevant research knowledge pertaining to our focal domain, a small review of social science, IS, CSCW and HCI literature related to online deliberation and e-democracy practices and system designs was undertaken. Citations relevant to the motivations of stakeholders involved in such practices were excerpted from the sources[5]. Of special concern were motivations having to do with information and opinion sharing, as our example's proposed technology – the collaborative filter – would primarily facilitate such practices and their projected consequences on stakeholder goals and motivations.

As shown in the below figure, *i**-related concepts were then identified within and among these citations, from stakeholder interviews (see section 3.3), and subsequently codified in *i** model syntax. Since we were consciously searching

[4] Note that the modeled scenario was not implemented in an actual usage environment; we merely utilize the context as a setting in which to demonstrate this modeling approach.

[5] Figures 1 and 2 visually depict the extraction and codification process.

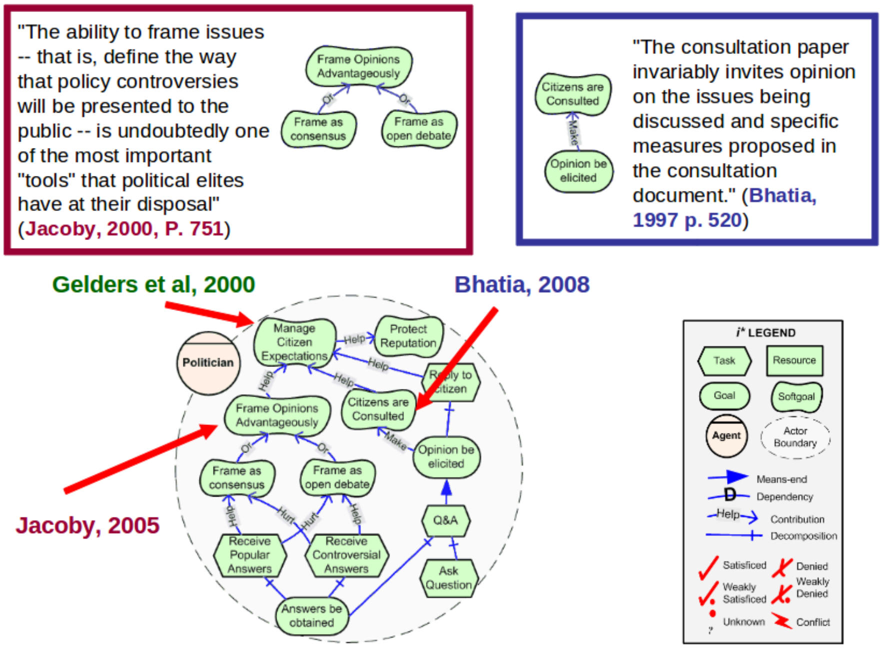

"The ability to frame issues -- that is, define the way that policy controversies will be presented to the public -- is undoubtedly one of the most important "tools" that political elites have at their disposal" (Jacoby, 2000, P. 751)
Frame Opinions Advantageously
Or
Frame as consensus
Frame as open debate
Citizens are Consulted
Make
Opinion be elicited
"The consultation paper invariably invites opinion on the issues being discussed and specific measures proposed in the consultation document." (Bhatia, 1997 p. 520)
Gelders et al, 2000
Bhatia, 2008
Jacoby, 2005
Politician
Manage Citizen Expectations
Help
Protect Reputation
Reply to citizen
Receive Popular Answers
Receive Controversial Answers
Hurt
Q&A
Ask Question
Answers be obtained
i* LEGEND
Task
Resource
Goal
Softgoal
Agent
Actor Boundary
Means-end
Dependency
Contribution
Decomposition
Satisficed
Denied
Weakly Satisficed
Weakly Denied
Unknown
Conflict

Fig. 1. Qualitative abstraction, codification and modeling of the politician actor

for and identifying stakeholder motivations, most extracted knowledge is expressed as *(soft)goals* and their interrelationships (expressed in the syntax as *dependencies* and *contribution links*.) The *actor* associated with the goal was also extracted, and all goals corresponding to that actor were associated in the model. Some intuitive connections between the modeled were needed in order to connect disparate elements. The researcher's domain knowledge played a large role in the identification of these correlations. In a sense, our role at this stage appears analogous to the knowledge management concept of 'knowledge intermediary' [26]; we identified relevant knowledge, represented and contextualized it so as to be useful to practitioners who could then utilize it towards some other end (eg: research-informed system design modeling).

Following this extraction and synthesis process, the planned functionality of the collaborative filter including two alternative configurations was modeled and intuitively linked to the various actors' goals. To do this, we assessed what the collaborative filter would require and what it would provide in the context described below. This permitted the evaluation of two simple alternatives in relation to their predicted contributions to stakeholder goals (see Figure 3 for the complete model).

3.2 Application to Climate Debate Community

Based on the literature review and research synthesis modeling method described above (respectively in sections 2.1 and 3.1), we modeled and evaluated several research-informed predicted effects of alternative collaborative filter configurations on the goals of real-world stakeholders. The application setting is an online community where users debate climate change issues and create plans on how to mitigate its effects.[6]

This demonstration example does not claim to be a complete synthesis of all relevant literature nor a full presentation of the setting's complexity. Specifically, we have simplified the number of actors, design features, goals and interdependencies in order to present a clear and concise summary of the modeling technique and evaluation method.[7]

3.3 Contextual Analysis: Stakeholders and Technology

Here we briefly outline some of the other requirements engineering activities (namely interviews with several stakeholders) we undertook in order to understand the application context and determine the type of relevant literature that would be beneficial to identify, extract and contextually synthesize.

The online climate community employs a discussion forum with multiple subtopics, interactive climate change mitigation plan creation tools based on a climate prediction model, and community-building features such as user profiles. Based on interviews with project staff, a major goal of the website is to become a resource for citizens to access and become informed about the various perspectives and plans that exist in the public sphere. These interviews motivated the literature search for resources about government-citizen communication in electronic environments, which we outlined above in section 2.1. Another identified need is to attract and retain users. Interviews with potential participants reflected what was found in the literature; some key goals include "feeling connected" to the issue and a sense that their participation has some kind of tangible effect.

Policy makers do not currently play an overt role in the website. However, as the preceding literature synthesis outlined, evoking a sense of citizen identity and providing a glimpse of a meaningful outcome is an important incentive for citizen communication.

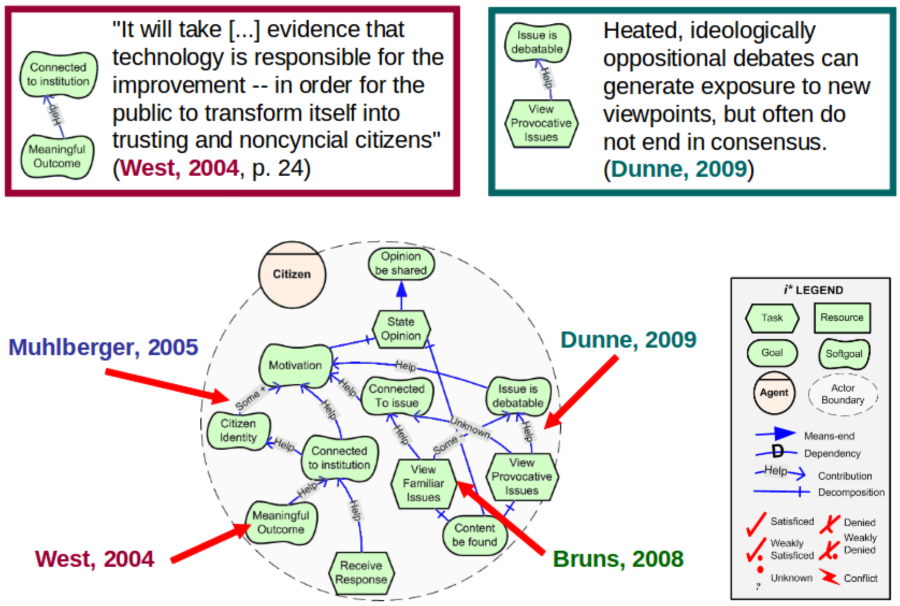
3.4 An *i** Model of the Climate Change Community

The modeled scenario (see Figure 3) supposes policy makers (simply modeled as `politician`) take on a more active role by asking questions to the community, who then may respond in a discussion forum.

A collaborative filter is proposed for introduction, in order to organize the resultant content. This technology has been chosen partially for demonstrative purposes and because the tool can leverage the collective opinion of the community and present salient and relevant information to the politician [14]. Additionally, this tool would be employed by users, who may easily respond to relevant or

[6] The climate community is loosely based on the Climate Collaboratorium [1].

[7] We expand on this note at the end of section 3.4.


"It will take [...] evidence that technology is responsible for the improvement -- in order for the public to transform itself into trusting and noncyncial citizens" (West, 2004, p. 24)

Heated, ideologically oppositional debates can generate exposure to new viewpoints, but often do not end in consensus. (Dunne, 2009)

Fig. 2. Qualitative abstraction, codification and modeling of the citizen actor

provocative posts. We do not model the technical details of the collaborative filter (the *how*), but only its functionality at a high-level of abstraction. If we were to 'drill down' into these technical considerations, it is likely that further, more concrete design alternatives might emerge. Our design alternatives are analyzed at the conceptual level; we look at *what* the alternatives functionally require and provide, as well as *why* the alternatives should be selected for inclusion in a design.

The $i*$ strategic rationale model of the setting (shown in its entirety in figure 3, below) attempts to answer the question "How does the configuration of a collaborative filter recommendation system affect the elicitation of citizen opinion by a politician?" We structured our approach to this question by modeling three actors: the `politician` (upper left area of the model) the `citizen` (upper right), and the `collaborative filter` (lower centre).[8] For simplicity of presentation, we depict a small number of interdependencies that we feel are representative of how a collaborative filter would interact with the *(soft) goals*, *resources*, and *tasks* of the other two actors. The modeled goals build upon the discussion of the literature in Section 2.1 and the model's construction followed the method described in the previous section.

[8] Technical artefacts may be modeled as actors according the the $i*$ syntax; as with human actors, a collaborative filter exists in interdependent relationships with others, and requires and provides certain resources in order to accomplish some goal.

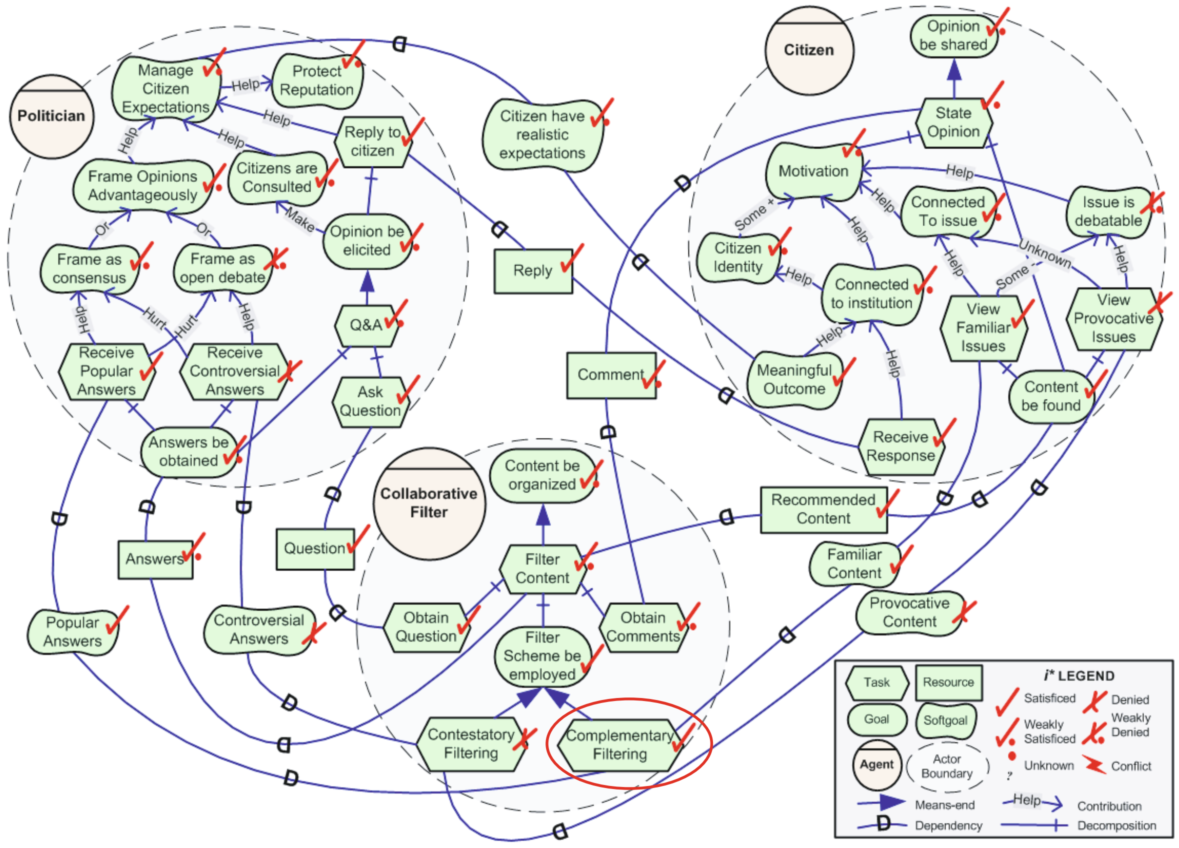


Fig. 3. Evaluating a Collaborative Filter Design Alternative for a Climate Change Debate Community

The model considers the two conceptual alternative configurations of the collaborative filter introduced in Section 2.1. Each are presented as distinct tasks, means of accomplishing the goal `filter scheme be employed` (shown in the lower area of the filter agent). The leftmost function intends to present users and politicians with contesting views and provocative content; it would be a `contestatory filter`. Alternatively, the technology could be configured as a `complementary filter`, whereby the filtered questions and citizen answers would be largely in accordance with user views. The model shows how the `collaborative filter` *depends* on the `politician` to provide questions (expressed here as a resource). It also depends on the `citizen` to provide comments.

The result of this filtering process is the creation of resources that the other actors depend upon. The `politician` receives `answers`, which may either be popular or controversial. Note the *dependencies* that link these two answer characteristics – modeled as a soft goal – to the alternative filter schemes – modeled as tasks. By following the *dependency* and *contribution* links propagating from these two design alternatives, an analyst may chart the path of each feature's contributions to stakeholder goals. A detailed explanation of this evaluative analysis is described in the next section.

Among other elements not modeled, the filter would also require some ranking data from citizen evaluation of comments in order to determine the criteria for filtering content, but we omit this consideration from to improve the model's legibility[9].

In summary, by being provided these resources, the collaborative filter may utilize one of two modeled design alternatives in order to filter these received comments and questions according to a certain scheme.

It is important to consider that design alternatives may impact individual citizen's goal achievement in variable ways; one individual may prefer to debate and another to discuss the familiar [28]. As such, careful requirements gathering, as mentioned above, should attempt to elicit the general feelings of the community and their proclivity towards debate and / or consensus.

3.5 Evaluating Alternatives

In addition to depicting strategic actor relationships and design alternatives, Figure 3 depicts the result of a qualitative evaluation procedure [20]. These results are represented by check marks attached to the *goals*, *soft goals*, *resources* and *tasks*. This evaluation methodology supports the iterative analysis of the effect of alternative choices upon stakeholder goals. In this evaluation scenario, the `complementary filter` is chosen (circled and check-marked in the model) while the design alternative – the `contestatory filter` – is not (marked with an 'X').

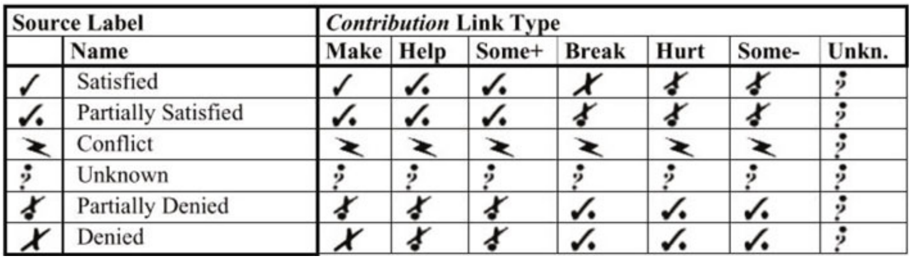
The effects of choosing the alternative propagate throughout the model via the values of dependency and contribution links originating from it. For example,

[9] An actual application of the modeling technique might require several models of the same domain, sliced into separate views of interrelated concerns in order to facilitate and simplify analysis [24].

since the `complementary filter` is selected in our evaluation scenario (and thus *satisfied*), the citizen may then view familiar issues which may *help* the citizen feel connected to the issue, but may have *some negative* impact on the amount of debate that can be generated. However, the citizen is *denied* reception of `provocative content`, which depended on the unselected `contestatory filter`. Since the contestatory filter is not selected, the recommendations do not directly help to encourage debate; the goal `issue is debatable` is *weakly denied.* Thus being `connected to issue` in turn helps to satisfy the citizen's goal, `opinion be shared`.

The `Politician` actor may similarly `receive popular answers`, dependent upon the `complementary filter`, which will help the actor to frame the issue as a consensus but *hurt* the ability to frame the issue as an open debate. As the alternative, `contestatory filter` is not employed, the `politician` will not `receive controversial answers`; thus nothing clearly helps to frame the issue as an open debate nor hurts the ability to frame the issue as a consensus. Thus, if the politician desires to present the discussion as having been a consensus, the `complementary filter` is the more attractive alternative to select.

Table 1. Evaluation Propagation Rules Showing Resulting Labels for Contribution Links. From [20].



Source Label		***Contribution* Link Type**						
	Name	**Make**	**Help**	**Some+**	**Break**	**Hurt**	**Some-**	**Unkn.**
✓	Satisfied	✓	✓•	✓•	✗	✗•	✗•	?
✓•	Partially Satisfied	✓•	✓•	✓•	✗•	✗•	✗•	?
⚡	Conflict	⚡	⚡	⚡	⚡	⚡	⚡	?
?	Unknown	?	?	?	?	?	?	?
✗•	Partially Denied	✗•	✗•	✗•	✓•	✓•	✓•	?
✗	Denied	✗	✗•	✗•	✓•	✓•	✓•	?

The model also shows that the `politician` may choose to reply to the community's answers. The `citizen` depends on the `politician` to present issues that generate realistic expectations and also to reply to his/her comments. If these are not fulfilled, then the citizen may not feel as connected to the political institution and thus motivation to contribute may suffer. This relationship is independent of the alternative configurations of the collaborative filter; yet it is instructive to include in the model to account for the reciprocal motivations underlying why a citizen might contribute.

Depending on the goals of the politician–whether the issue is framed as an open debate or a consensus–the designer may select the alternative that contributes most beneficially towards the accomplishment of those goals.

Through the above example, we have demonstrated how the i^* modeling technique can support reasoning about how configuration choices of the collaborative filter would impact high-level stakeholder goals. Furthermore, we have outlined the basis of a method for retrieving and extracting domain knowledge from relevant scholarly literature and synthesizing it with knowledge from the application environment derived from stakeholder interviews.

These methods are fairly context-independent. Depending on how well-defined the problems within the application context are and/or the degree of domain knowledge on the part of the design team, the balance between traditional and research-oriented requirements modeling could shift. Nevertheless, the research modeled in our climate change example could likely be re-purposed and re-contextualized for many other government-citizen communication situations; particularly in group decision-making or online deliberation environments where similar issues of motivation, exposure to competing ideas and connection to actual policy making would be prevalent.

4 Discussion and Future Work

We have shown how the demonstrated model construction method can produce models that can help organize and refine contextually situated research. Early results from our investigation into the practicality of this method have provided useful and encouraging feedback.

We obtained initial input from interviews with six designers of online deliberation systems. The designers indicated that a goal-oriented approach to codifying this domain's design knowledge can have multiple uses and benefits to system designers. Early results indicate that a small sample of designers in this area do not consult the academic literature related to their field; several participants expressed frustration regarding the length of publications and the corresponding time investment required to obtain relevant information. These individuals have found that a structured representation of their domain's information in terms of goals to be achieved through implementable design features is very relevant and useful to their design work.

Nevertheless, the method we have described does little to address this domain's need for collaborative design knowledge sharing [8] in order to effectively and efficiently support the design activities of practitioners.

We believe our method can be improved to better support design knowledge reuse. One area of improvement is the literature review and codification method, that was conducted informally with a specific case in mind, and not preserved for later retrieval. This process could be tailored towards a systematic means of repositing design knowledge for later contextualization in an application environment. The method could also be more clearly defined and hence more reproducible. A reflection on these limitations coupled with insightful comments by peers and reviewers has inspired the ongoing development of a more robust means of engaging with domain knowledge and creating research-informed early requirements models for social media in support of citizen-government communication, which we next address.

Scholarship and design practice in this domain could be improved if its practitioners were able to better leverage each other's research findings and critically build on past results [8]. As such, designers of social media platforms intended to support deliberative activities such as turn-taking discussion of climate change issues, iterative voting, or structured argumentation stand to benefit from such knowledge sharing. To address such issues, information systems designers, human factors specialists, and design studies scholars have presented methods and knowledge repositories that intend to support the reuse of past design knowledge. Adapting their work to the domain of social media-supported online deliberation could theoretically provide a path towards a solution to the above concerns.

Based on the above motivations, we are currently iteratively developing a framework for the systematic codification, retrieval, representation, and recontextualization of design knowledge for social media. Our two primary framework elements are based on Hevner et al's [18] division between disciplinary and environmental information systems design knowledge.

The first element of our framework is a knowledge base derived from online deliberation-related design literature, conceptualized based on domain concepts and structured primarily according to elements from requirements engineering and design science theories. Second is a methodology for analyzing this knowledge base through incremental association of contextual details with knowledge base items. By combining these two strands, we aim to create a robust, model-supported framework which system designers may utilize in order to reflect, recontextualize, and build upon previous scholarly and practitioner findings from designing for this domain.

We are currently undertaking an empirical study that will examine the validity of our proposed framework. As mentioned in the discussion, interviews as well as tool demonstration sessions with practitioners who design online social media for government-citizen communication and/or group decision making and deliberation are being conducted. Results from these sessions are helping us to refine and better align our framework with contextually-situated design practices.

5 Conclusions

This paper demonstrates an application of goal-oriented analysis techniques to support reasoning about design alternatives. The application example considers citizen opinion elicitation in the context of the climate change debate. Varying configurations of a social medium – a collaborative filter – were analyzed and evaluated based on the goals derived from the literature.

By synthesizing relevant research literature with traditional requirements gathering and analysis methods, as well as adopting a goal-oriented view of the domain designers may build upon and recontextualize past knowledge that may provide additional, reputable perspectives to the various design alternatives under consideration. By analyzing the impact of such configurations at an early requirements determination stage, the designer may ensure that the implemented system satisfies stakeholder goals more effectively than various alternatives.

Acknowledgments. We thank Arlo Murphy for proofreading the abstract. Karen Louise Smith and Lysanne Lessard have also provided valuable feedback. We are grateful to the anonymous reviewers and participants of the MSM '10 conference who were instrumental in helping us identify future directions for the project. Financial support from the Natural Sciences and Engineering Research Council of Canada is gratefully acknowledged.

References

1. The climate collaboratorium, http://www.climatecollaboratorium.org
2. Ahmed, S.: Encouraging reuse of design knowledge: a method to index knowledge. Design Studies 26(6), 565–592 (2005), http://linkinghub.elsevier.com/retrieve/pii/S0142694X05000177
3. Bhatia, V.K.: Democratizing government decision-making: A study of public discourse in Hong Kong. Journal of Pragmatics (1997)
4. Bruns, A.: Life beyond the public sphere: Towards a networked model for political deliberation. Information Polity 13(1), 71–85 (2008)
5. Chen, H.: AI, E-government, and Politics 2.0. IEEE Intelligent Systems 24(5), 64–86 (2009)
6. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-functional requirements in software engineering (2000)
7. Dardenne, A., Van Lamsweerde, A., Fickas, S.: Goal-directed Requirements Acquisition. Science of Computer Programming, 3–50 (1993)
8. Davies, T.: The Blossoming Field of Online Deliberation. In: Davies, T.R., Peña Gangadharan, S. (eds.) Online Deliberation: Design, Research, and Practice, pp. 1–20. CSLI Publications, San Francisco (2009)
9. Deaton, B., Lintner, A., Harrington, D.: Evaluating an Environmental Right: Information Disclosure, Public Comment, and Government Decision Making in Ontario. Canadian Journal of Agricultural Economics/Revue Canadienne D'Agroeconomie 56(3), 277–294 (2008)
10. Donzelli, P., Bresciani, P.: Goal-Oriented Requirements Engineering: A Case Study in E-government. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, pp. 601–616. Springer, Heidelberg (2003)
11. Dunne, K.: Cross Cutting Discussion: A form of online discussion discovered within local political online forums. Information Polity 14(3), 219–232 (2009)
12. Esfahani, H.C., Yu, E.: A repository of agile method fragments. In: Münch, J., Yang, Y., Schäfer, W. (eds.) ICSP 2010. LNCS, vol. 6195, pp. 163–174. Springer, Heidelberg (2010)
13. Esfahani, H.C., Yu, E., Cabot, J.: Situational Evaluation of Method Fragments: An Evidence-Based Goal-Oriented Approach. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 424–438. Springer, Heidelberg (2010)
14. Freyne, J., Jacovi, M., Guy, I., Geyer, W.: Increasing engagement through early recommender intervention. In: Proceedings of the Third ACM Conference on Recommender Systems, pp. 85–92. ACM, New York (2009)
15. Gelders, D., Cock, R.D., Roe, K., Neijens, P.: The opinion of Belgian government communication professionals on public communication about policy intentions: Pros / cons and conditions. Communication 23(Government Information Quarterly), 281–292 (2006)

16. Goodin, R.E., Dryzek, J.S.: Deliberative Impacts: The Macro-Political Uptake of Mini-Publics. Politics & Society 34(2), 219–244 (2006)
17. Grönlund, K., Strandberg, K., Himmelroos, S.: The challenge of deliberative democracy online A comparison of face-to-face and virtual experiments in citizen deliberation. Information Polity 14(3), 187–201 (2009)
18. Hevner, A., March, S., Park, J., Ram, S.: Design science in information systems research. Mis Quarterly 28(1), 75–105 (2004)
19. Hilts, A., Yu, E.: Modeling social media support for the elicitation of citizen opinion. In: Proceedings of the First International Workshop on Modeling Social Media, pp. 1–4. ACM, New York (2010)
20. Horkoff, J., Yu, E.: A Qualitative, Interactive Evaluation Procedure for Goal- and Agent-Oriented Models. In: CAISE Forum. CEUR-WS.org, vol. 453, pp. 19–24 (2009)
21. Jacoby, W.: Issue framing and public opinion on government spending. American Journal of Political Science 44(4), 750–767 (2000)
22. Kavanaugh, A., Kim, B., Perez-Quinones, M., Schmitz, J., Isenhour, P.: Net gains in political participation: secondary effects of internet on community. Information, Communication and Society 11(7), 933–963 (2008)
23. Kolsaker, A., Lee-Kelly, L.: Citizens' attitudes towards e-government and e-governance: a UK study. International Journal of Public Sector Management 21(6-7), 723–738 (2008)
24. Leica, M.F.: Scalability concepts for i* modelling and analysis. Master's thesis, University of Toronto (2005)
25. Lock, S., Shapiro, R., Jacobs, L.: The impact of political debate on government trust: reminding the public what the Federal government does. Political Behavior 21(3), 239–264 (1999)
26. Markus, M.: Toward a theory of knowledge reuse: Types of knowledge reuse situations and factors in reuse success. Journal of Management Information Systems 18(1), 57–93 (2001)
27. Moayerzadeh, A.: A Goal-Oriented Representation of Service-Oriented Software Design Principles. Master's thesis, University of Toronto (2008)
28. Muhlberger, P.: Human agency and the revitalization of the public sphere. Political Communication 22(2), 163–178 (2005)
29. Narro, A.J., Mayo, C., Miller, A.F.: Legislators and constituents: Examining demographics and online communication tools. Information Polity 13, 153–165 (2008)
30. Oxman, R.E., Planning, T.: Precedents in design: a computational model for the organization of precedent knowledge. Design Studies 15(2), 141–157 (1994)
31. Park, H., Choi, S.: Focus Group Interviews: The Internet as a Political Campaign Medium. Public Relations Quarterly 47(4), 36–43 (2002)
32. Pastor, O., Estrada, H., Martínez, A.: Strengths and Weaknesses of the i* framework. In: Yu, E., Giorgini, P., Maiden, N., Mylopoulos, J. (eds.) Social Modeling for Requirements Engineering, pp. 607–643. MIT Press, Cambridge (2011)
33. Renton, A., Macintosh, A.: Computer-supported argument maps as a policy memory. The Information Society 23(2), 125–133 (2007)
34. Thaler, R.H., Sunstein, C.R.: Nudge: Improving Decisions About Health, Wealth, and Happiness. Yale University Press, New Haven (2008)
35. Werneck, V.M.B., Oliveira, A.d.P.A., Leite, J.C.S.d.P.: Comparing GORE Frameworks: i-star and KAOS. In: Workshop em Engenharia de Requisitos (WER 2009), Val Paraiso, Chile (2009)

36. West, D.: E-government and the transformation of service delivery and citizen attitudes. Public Administration Review 64(1), 15–27 (2004)
37. Wieringa, R.: Requirements engineering: Problem analysis and solution specification (Extended abstract). In: Koch, N., Fraternali, P., Wirsing, M. (eds.) ICWE 2004. LNCS, vol. 3140, pp. 13–16. Springer, Heidelberg (2004)
38. Yu, E.: Social Modeling and i*. In: Borgida, A., Chaudhri, V., Giorgini, P., Yu, E. (eds.) Conceptual Modeling: Foundations and Applications: Essays in Honor of John Mylopoulos, pp. 99–121 (2009)

Grooming Analysis
Modeling the Social Interactions of Online Discussion Groups

Else Nygren

Uppsala university, Department of Informatics and Media,
Box 513, 75120 Uppsala, Sweden
else.nygren@im.uu.se
http://www.uu.se

Abstract. This chapter looks into different ways of analyzing and modeling the time dependent development of social interactions between members in online discussion groups. Social grooming is an activity in which individuals bond and reinforce social structures. To groom someone can be for instance to mention someones name in a discussion or to cite something said by that person. The number of grooms received by a group member can be analyzed and used as an indicator of the social status in the group. By performing grooming analysis it is possible to attain information about how the relative status of the group members changes over time. The data in this study is taken from two different international online discussion groups. A validation showed that the estimate of status based on the grooming analysis showed remarkable correspondence with the collective status ranking performed by a group of independent evaluators.

Keywords: Social media, social network, community, social status, social dynamics, group dynamics, gender, Mathematica.

1 Introduction

At international online discussion sites, people from around the globe can exchange knowledge and opinions and become virtual friends. Even if these people never meet face to face, it seems that social hierarchies develops also within such groups. Some get a lot of attention whereas others get almost none. This is interesting since the medium does not give away any of the clues we are used to interpret in real life. We know that in real life, appearance, voice, accent as well as body language affect how we appraise other people and thus contribute to the social rank structure in a group. None of these clues are available in a virtual group. Because of this it is interesting to know what factors are influencing the social structuring in virtual groups.

Of particular interest is the notion of social status ranking of members in the group. This can be seen as a macro structure or a property of the group that emerges out of the many social interactions between the group members. We can say that the micro behavior of the group members give rise to the macro property of social stratification of the group. To be able to study such phenomenon we need to model the development over time in the group.

The approach taken here is to look at the discussion group from a psychological perspective and to search for signs of attention in the posts made. This is smart because it makes it possible to get many more insights into the social dynamics over time of the group, compared to only analyzing the number of posts that people make. Also it makes it possible to construct a social network graph even for groups that are not making links to each other.

2 Background

Social Interaction and the Emergence of Status in Groups. Social status replicates itself across different social circumstances. An important distinctions is made between *maintenance of status* and *emergence of status*. Maintenance of status occurs when the group members already are ranked. Emergence of status however is what happens when individuals that do not know each other come together. Processes of social interaction between the members result in the emergence of status rank in the group [1]. High status members of a group can be characterized by being helpful, remind others about rules, provide suggestions and establish norms for the group. Furthermore they are not targets of aversive behavior but they will defend themselves when attacked. Low status members of a group can be either rejected members or neglected members. Rejected members are characterized by aggressive and disrupting behavior, they start conflicts, often approach others and are rejected, thereby collecting a lot of rejection experiences. They also respond to aggression by more aggression. Neglected members on the other hand are characterized by being shy and getting very little attention from the other members, they also respond to aggression by withdrawal, ignoring the aggression or by submission without retaliation. [1].

Social Grooming. In social animals, grooming is a major social activity, and a means by which animals that live in proximity can bond and reinforce social structures, family links, and build relationships. Social grooming is also used as a form of reconciliation and a means of conflict resolution in some species. In animals, it has been studied how grooming was performed by one animal upon another animal of the same species. It was found that grooming was significantly correlated with social rank. High-ranking individuals received more grooming than their low-ranking group mates. The results support the notion of grooming as an indicator of social status [2].

Social Attention Analysis. For humans the notion of giving attention has been used as a parallel to grooming in animals. In social attention analysis the objective is to measure the attention that people give to each other over time. To do this a particular metric is used. This metric keeps track of who give attention to whom at what time. By analyzing such data it is possible to assess the communities attention to a group member, the group members impact over time and the group members diversion of attention over time [3]. This method requires that the notion of attention can be defined as something measurable. When social attention analysis is carried out in face-to-face settings such measures can be for instance nodding in approval when someone is speaking or reciting what another group member has said previously.

Social Status in Online Groups. In a face-to-face situation there are a number of signs that can be used to assess status. Examples are erect posture, glares, eye contact and assertive speech. In online discussion groups there are no such signs, thus status has to be assessed based on the information available in the discussion group pages. So how is status assessed in an online setting?

Sometimes, a social media group is started by a small group of friends and then grows with the addition of their real-world friends. In that situation the social status pattern is reproduced from the real-world status.

If a social media group is evolving over time by the accumulation of users that does not formerly know each other we have a situation where the social status pattern emerges over time in a group of mutual strangers independent of any former real-world status patterns. What can be the basis of such emergent status patterns?

In groups that work together, the number of contributions and the quality of contributions can be a basis for assessment of status. In many online groups there are links between the group members. If the numbers of links are visible to the group members it is possible that this could be used as a basis for the assessment of status. In groups that are merely socializing, the assessment can be based on how interesting a particular group members posts are. Another possible base for assessment of status is how much a group member is socially attentive to the other members of the group. Thus a high-status member would be helpful, encouraging and pay attention to others in the group.Other basis for assessment of status can be gender, geographic location or displayed knowledge about a particular topic.

There are thus several possible things that can work as a basis for the assessment of status in an online group and there is probably not a single scalar rank in the group. Regardless of what basis is used for the assessment of status in an online group there is a real social status out there in the sense that we could ask people about the social status of group members and they will give you an answer. This answer is dependent on what special meaning that individual gives to the concept of social status. In this work we define social status in an online group as the collective ranking made by a group of individuals that possibly base their ranking on different factors. In this work, we want to find an indicator of social status that is possible to assess by analyzing the available content.

The question is thus: - Can we find an indicator of social status in the available data from an online discussion group - Is the social status ranking obtained by this indicator equivalent to the collective social status ranking as perceived by a group of individual evaluators. If we can find such a measure it will be possible to analyze how the status emerges over time and evolves dynamically.

Methods for Analyzing Social Status and Group Dynamics Online. Different types of methods have been used to assess the social status in online groups (the concept of status used here is similar to the concepts of *influence* or *reputation*). The methods use different types of indicators of status. These are chosen dependent on what kind of data that is available and the purpose of the study. The purpose is most often to find the most influential persons in the group with a possible sub-goal of exposing these people to marketing activities.

One commonly used indicator of status is the number of user contributions. Here it is assumed that the more a person contributes to the group in the form of posts, uploaded material and answers to questions, the higher the social status of the individual. Yan and Vassileva used a systems dynamics approach to study the incentive mechanisms in a virtual community [4]. In this work a combined measure of different forms of user participation was used as an indicator of social status. They successfully modeled the dynamics of how members raised to higher and higher status in the group as measured by their status class that was assigned to them by the system software.

Another, subtler, indicator of status is the *quality* of the contributions. Agent based methods models the essential characteristics of the individual, as well as the rules and the global consequences of the interactions between individuals. Zhang and Tanniru made an agent based study where the characteristics included expertise level, activeness level, sharing level and social gain. In this study it was assumed that the quantity and quality of the messages posted was an indicator of social status (reputation) [5].

Another indicator of status that has been used is *the number of in-links* to a person. Agarwal et al developed a model for identifying the most influential bloggers in community blogs. In their model, a weighted measure combined by number of comments, number of in- and out-links and length of the post was used as an indicator of social status (influence). One thing that was apparent in their study was that activity was not correlated to influence. There were bloggers with low activity that were nevertheless very influential and vice versa [6].

Some virtual communities use *peer ranking of the quality of messages* to recognize and label an agents reputation. This has been used as a validating indicator of status [6].

Another indicator that has been used in settings of online marketplaces is *the quality of former interactions.* Sabater and Sierra have constructed a model of social status (reputation) that generates sociograms that show the relations between individuals. In their work the quality of the former social interactions in a marketplace system was used as an indicator of social status (reputation) [7]. Within the field of bibliometrics, *the number of citations* that authors of scientific publications receive over time has been used to indicate status [8]. Many interesting results have been found regarding the notion of status and its relation to a tree structure of citations of citations. Here, the total number of citations is used to indicate status.

In the studies mentioned above, the validations made were based on other computer-generated data. There has not been any study where validation was made in terms of comparison with status as appreciated by a group of humans.

Many of these indicators have no relation to social interactions in a group. This type of status can only be observed by a global system having access to, for instance, all citation data and we cannot know if this is equivalent to emergent social status as attributed to a person by a group of people. This has been noted by Sabater and Sierra who writes *"This external position gives the analyst a privileged watchtower to make this analysis"*[7]. They conclude that, actually, each agent has to do this analysis from its own perspective.

Our approach to analyzing social status is to try to find an indicator that can be shown to be equivalent to the social status ranking that emerges from many individual judgments by individuals each assessing a limited set of interaction data. In the study described here the concept of *social grooming* is used as an indicator of status. The social status ranking based on this indicator will be validated by comparison to the global status ranking that emerges from the combination of individual assessments of status made by a group of people that each have access to a limited set of the data. The perspective chosen is thus to study how micro activities by the group members together generates macro properties of the group, more specifically the macro property of social status stratification.

3 Empirical Material

The two online discussion groups were selected for two reasons. First I wanted to have two groups of similar sizes in terms of number of people and number of posts. Secondly I wanted to have groups that were different in terms of content since I wanted to find out what a content independent analysis can reveal about a group. Also, though not a member, I happened to be familiar with both groups. (Please note that *subject* in this text refers to a person in accord with tradition in experimental psychology)

The two data sets consisted of 1604 and 971 posts respectively. Data was captured by copying the content from the website pages and saving as text files. The data represented the activity in the groups over a period of 3-4 years. There were no links between group members. See Table 1 for more information about the data sets.

Online Discussion Group 1: Fashion. The first of the groups studied is the people that comment about fashion in a particular thread on the site *Fashionspot*. The website is dedicated to discussions of different aspects of design and fashion. This particular thread is about a rock band with a distinctive style in fashion. The discussions are about the style and appearance of the band members and their girlfriends but also about the music of the band. The activities that the group members engage in are viewing posts, commenting, posting pictures or links, asking questions, answering questions and generally socially chatting with each other. The data structure was as follows: Date and Time of Posting, Post Number, Subject Name (a unique name was required to be able to post comments, this was chosen by the subjects themselves), Subject Location (this text field was voluntary and could be filled in with city or country, or sometimes only by continent), Subject Gender (this was voluntary, but most filled this in), Quote (the name of the quoted subject was automatically filled in when one subject quoted another subject), and a Text Field (which contained the written text).

Online Discussion Group 2: Science. The second of the groups studied is the people that comment about computer science in a particular section called *The core science of simple programs*on the forum *A New Kind of Science.* The discussion regards cellular automata and computational theory. The activities that the group members engage in are viewing posts, commenting, posting program files, pictures or links, asking questions, answering questions and to some extent, socially chatting with each other. The data structure was as follows: Date and Time of the Post, Subject Name (a unique name was required to be able to post comments, this was chosen by the subjects themselves), Subject Affiliation (the subject could give an affiliation as a company or a university), Subject Location (this text field was voluntary and could be filled in with city or country), and a Text Field containing the written text.

4 Method

4.1 Analysis of the Posting Activity

The text files were imported to Mathematica. A program was written that searched for text patterns and picked out data and reconstructed the following general descriptive data for each post: Post number, Post Day (this was calculated so that the first day of posting was day 1, the next day, day 2 and so on) and Subject number (this was constructed so that the first poster was given number 1, the next poster number 2 and so on).

For each subject data about gender and location was picked out if this was available otherwise this was categorized as unknown. The gender data in the Science group had to be inferred from the names of the subject since no registration of gender was made. The gender classification was made by identification of a male or a female name. All names that were ambiguous were classified as unknown. For each given location a table was constructed that stated if the location could be described as an English speaking country or not.

4.2 Grooming Analysis

The idea behind the groom analysis is to combine two methods used in other contexts when trying to map out the social rank order in a group. The first method is simply that used by primate researchers of counting number of grooms and relating that to the social rank order in the group. The second method is social attention analysis, which tries to capture how a group gives differentiated attention to the different group members. In the groups studied in this work, the structure is like a bullentin board with every post basically directed to all the members of the group. If a post contains a name reference to one of the other members then this is a sign that this individual has been seen and is acknowledged. In the studies by Joyce & Kraut it was found that replies to posts increased the probability that a user would post again. This was so regardless of the emotional tone in the answer. Because of this we argue that the name reference can be seen as an indication of that a member receives attention in the group [9]. Attention can in this context be seen as a social reward. Two assumptions are made. First it is assumed that attention from other group members works as a social reward and thus is a positive reinforcer of participating in the group. Secondly, it is assumed that the amount of attention that an individual gets in the group is an indicator of the social status or rank order in the group. In particular, a person who gets more attention than he or she gives to others is considered to be a high-status person while a person who receives much less than he or she gives to others is considered to be a low-status person. It is thus assumed that this balance reflects the social status in the group. By finding indicators of attention in the data, and keeping track of who gives to whom, it would thus be possible to quantize an estimated measure of an individuals status in the group.

In this analysis the concept of groom is defined as a social reward, in form of attention from other group members. For the Fashion group, attention was operationalized as one of two things. The first was *explicit mentioning of someones name* in a post. The second was *quotation* of something that the group member had written in a post. For the Science group, attention was operationalized in a similar way as one of two things. The first was *explicit mentioning of someones name* in a post. The second was *commenting on something that a group member had posted.* These acts were possible to identify in the data. If a group member received attention in this way, it was said that the he or she was groomed, or received a groom. For each identified act of grooming, it was registered who gave the groom and who received the groom. A groom balance was constructed for each subject. This increased by one each time the subject was groomed, and was reduced by one each time the subject groomed someone else. In this way we could construct a time-dependent groom-balance for each subject. Also the accumulated number of grooms received for each subject was counted. The data was imported to Mathematica and a program was made to pick out grooms in all of the posts. Unfortunately, the mentions of some names had to be manually edited because of short forms or misspelling of the names of members. This was the case in 4% of the grooms in the Fashion group. For each member the program kept track of the number of posts made, the number of grooms given to others and the number of grooms received from other group members.

Table 1. Results of the Analysis of the Posting Activity

Basic Data	Fashion	Science
Number of active group members	187	167
Number of posts	1604	971
Number of days studied	1189	1604
Gender distribution: Females	84,1%	4,2%
Gender distribution: Males	8.5%	79.0%
Gender distribution: Unknown gender	7.4%	16.8%
Group members from English speaking countries	36.4%	35.5%
Group members from Non-English speaking countries	19.3%	14.5%
Group members from unknown location	44.9%	50.0%
Countries represented	Argentina Australia Belgium, Brazil Canada, China Colombia Czech Republic El Salvador England Finland, France Germany Israel, Kenya Latvia Netherlands Norway Philippines Poland, Russia Scotland Spain, Sweden Turkey USA	Australia Brazil Canada England Egypt France Germany India Israel Italy Korea Mexico Netherlands New Zealand Paraguay Romania Switzerland Vietnam Ukraine USA
Derived Measures	Fashion	Science
Posts per subject all	8,6	5,8
Posts per subject English-speaking countries	10,0	6,6
Posts per subject Unknown location	8,8	6,1
Posts per subject Non-english-speaking countries	4,7	3,1
Posts per subject Females	9,7	1,7
Posts per subject Unknown gender	2,8	6,1
Posts per subject Males	2,2	6,0

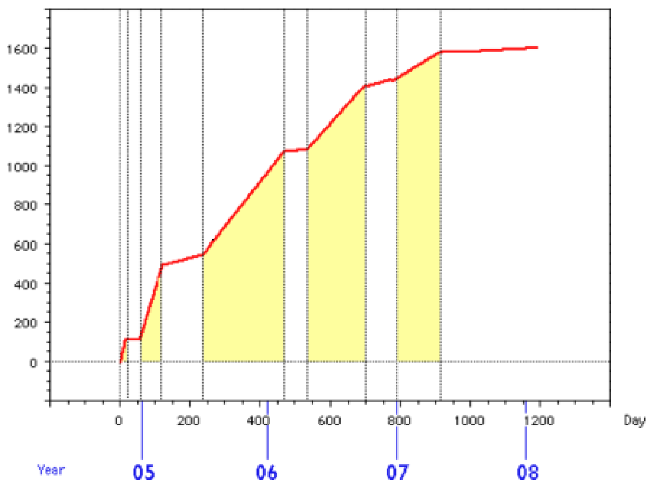


Fig. 1. Fashion Observed data. Accumulated number of posts as a function of days. Periods of high activity are interspersed with low activity.

5 Results

The basic results of the analysis of the posting activity is presented in Table 1. By plotting each post as a data point where the day of the post is shown on the x-axis and post number on the y-axis we can get an overview of the posting activity over time. The height of the curve shows the accumulated number of posts. The slope of the graph shows how the activity in the group varied over time. The graph shows that periods of intense activity (high slope) intermingle with periods where the activity in the group almost died out. This kind of graph can also reveal variation coupled to seasons, however in this example no such tendency can be seen. See Figure 1.

Group Member Activity over time. In this graph each post is shown as a data point where the post number is shown on the x-axis and subject number on the y-axis. By doing this plot we can get an overview of the posting activity by different group members. Figures 2 and 3 show the group member activity graph for the Fashion and the Science groups respectively. The slope of the graph shows us how the number of members increases over time. We can also see that some members post many times and keep posting all the time, while many post only once. If we compare the graphs for the two groups we can see some differences. In the Fashion group there are no group member that stays in the group the whole time, in fact if we compare the first and the last quarter the posting members are a completely different set of people. In contrast we can see

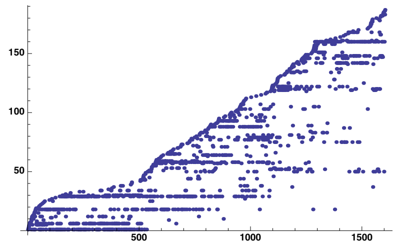


Fig. 2. Activity Graph Fashion

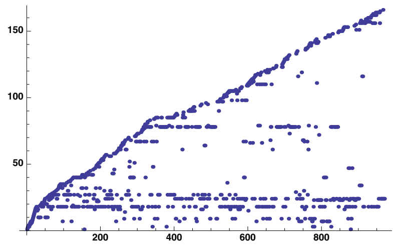


Fig. 3. Activity Graph Science

that in the Science group a few members keeps posting steadily throughout the observed time period of more than four years, however many in this group post only once.

Differences in Activity. In figures 4 and 5 we can see an ordered plot of the number of posts made by individual subjects for the two groups. These plots show that there is an uneven distribution of posts. A few subjects make most of the posting and a large number of subjects only make one post. The two groups are very similar in this respect. Naturally, we can expect a difference between different subjects since the persons joined the group at different times and thus have not had the same effective amount of time available to post comments. Regression analysis of the number of posts as a function of the day of entering the group, show that indeed there is a small effect. About 4% of the variance can be explained by this correlation.

The activity can be broken down in gender and language groups. The under-represented gender post much less per individual, about 2 per person, than the majority gender group, about 6-10 per person. People from English-speaking countries makes about twice as many posts as people from non-English speaking countries.

Results of the Grooming Analysis. Let us first look at the Fashion group. The overall occurrence of mentions was 152 and the occurrence of quotes was 295 in the data set. The concept of groom, defined as the sum of quotes and mentions is thus 447 in the whole set. This corresponds to one mentioning for every 10,6 posts and one quote for every 5,4 posts.

In the Science group the number of grooms were 816 composed of 342 mentions and 474 comments on posts. Thus there was one mentioning for every 2,8 posts, and one comment for every 2 posts. This means that there were relatively more grooming in the Science group compared to the Fashion group.

Of the 187 people in the Fashion group, 113 or 60% got no grooms at all. Of the 74 people that received any grooms there was a center group of 8 people that received 50% of the grooms. At the very center was four females that got 40% of the total number of grooms. 2 of those were from US, one from Canada and one did not disclose her location.

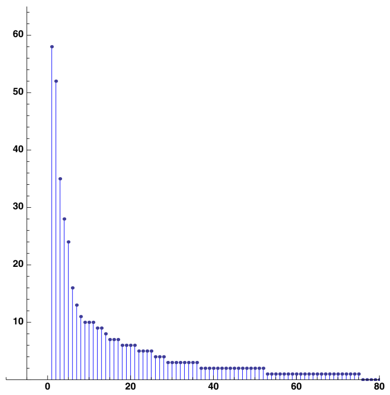


Fig. 4. Distribution of posts Fashion

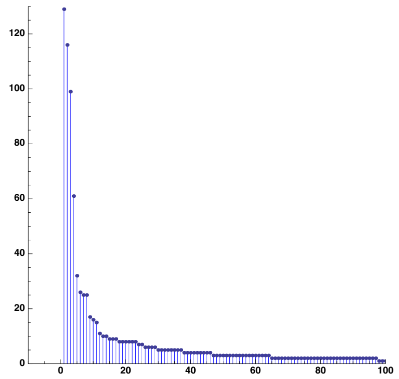


Fig. 5. Distribution of posts Science

Of the 166 people in the Science group, 46 or 27,7% got no grooms at all. Of the 120 people that received any grooms there was a center group of 10 people that received 50% of the grooms. At the very center was five males that got 38,5% of the total number of grooms. 3 of those were from US, two did not disclose their location.

The correlation between number of posts and number of grooms for each subject was 0,94 for the Fashion group and 0,03 for the Science group. The mean correlation for the two groups was 0,48.

Reciprocity in the Distribution of Grooms. A calculation was made of the relation between grooms given and grooms received. (Fashion 96 people gives nothing, 19 or 19,8% of these get groomed, 91 people gives and 55 or 60,4% of these were groomed). (Science 56 people gives nothing, 37 or 66% of these get groomed, 110 people gives and 83 or 75,5% of these were groomed). Thus, in the Fashion group, your chances to get groomed increase from 20% to 60% if you groom someone else. In the Science group, the chances increase from 66% to 76%. The effect was not statistically significant (ANOVA P-value 0.410992), but the data indicates that it is generally a good idea to groom someone if you want to be groomed yourself.

Effect of Grooming on Posting Activity. For the Fashion group: 187 people made a first post, 28 of these got feedback in the form of a groom. Of the 28 persons that got feedback, 20 made at least one more post. Of the 161 persons that did not receive any feedback at all, 43 made at least one more post. Thus the probability of making a second post was 0,72 for those who received grooming and 0,27 for those who did not. The mean number of posts needed before any grooming was received was 3,3. The maximum was 17 posts. That means that one person made 17 posts before anyone acknowledged her presence in the group. Most of those who were groomed, however, was groomed after one or two posts.

In the Fashion group, 15,5% of the females were groomed after their first post compared to 5,9% of the males. The females got 1,13 times their input whereas the males only got 0,37 times their input. So actually the males got much less return for their investments than the females in this group. In the Science group, 52% of the males got feedback after their first post, compared to 33% of the females. In the Science group, both the males and the females got 1,0 groom for every invested.

In the Fashion group, people from English-speaking countries got 1,0 for each groom while the people from non-English speaking countries got 1,76. In the Science group, people from English-speaking countries got 0,94 for each groom while the people from non-English speaking countries got 1,10. Thus it seem that the reason that people from non-English speaking countries get fewer grooms is not that they get less return on their investments, bur rather that they give less grooms in the first place.

The nicknames used by the subjects in the two groups could be classified in three sets. Nicknames that were actual names, nicknames that were not names but pronouncable words and nicknames that were combination of letters and numbers and not possible to pronounce. We thus have the three categories of nicknames: Names, Words and non-pronouncable. Names got most, pronouncable words next and unpronouncable nicknames got least. (Fashion: 2,96, 2,04, 0,8; Science: 1,31, 0,60, 0,33). This means that if you want to be groomed you should avoid an unpronouncable nickname.

The Grooming Network. The notion of grooms are directional in the sense that it is possible to identify that one subject gives a groom to another subject. This means that it is possible to construct a grooming-network showing the social interaction between the group members. Each subject is considered as a node in this network and the degree of social interaction can be visualized as the connection strength. The community modularity was 0,55 for the Fashion group and 0,44 for the Science group. As the groom data holds information about when a particular groom was given it is also possible to construct the grooming networks for different time periods, an example of a the grooming-network at a particular point in time can be seen in figure 6.

The Groom Balance. Every groom can be said to represent a certain point in time. For every post including a groom it is possible to calculate the groom balance for each subject. The idea behind the groom balance is that for a random member of the group we can expect reciprocity so that on average people give approximately as many grooms to others as they receive themselves. A person with many more grooms received than given would indicate that that person had a high status at that particular moment in time. The groom balance was calculated for each subject in the folllowing way: the number of grooms the subject had received minus the number of grooms a subject had given to others.

In figure 7 we can see the groom balance over time for a subject in the Fashion group. This is an example of a person who gives about as much as she get and the groom balance consequently fluctuates around the zero level.

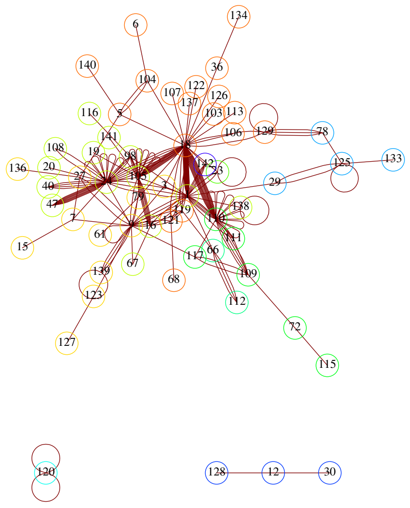


Fig. 6. Science. The Grooming Network show the social interactions. Every line represents a groom from one member to another.

In figure 8 we can see the groom balance over time for a subject from the Science group. This is an example of a person that gives a lot but receives very little. It is argued that this as a sign of low status.

In figure 9 we can see the groom balance over time for a subject from the Fashion group. This subjects receives more than what is given to others. It is argued that this as a sign of high status.

It also possible to plot the groom status for the whole group. In figure 10 we can see for the Fashion group how different high status individuals replace each other successively over time. When one high status member leave the group, it seems that another member soon takes her place as the alpha member in the group.

6 Validation

The grooming analysis resulted in a value of total number of grooms received for each subject. It is assumed that this value can be used as an estimate to rank order the group members according to their social status in the group. To know if this estimation corresponds to reality it is necessary to make a validation. But how can we get a real measure of the social status of the group members? One way would be to let someone make subjective judgements and rank order the members of the group. This is problematic since it would only reflect the opinion of one single individual and therefore we rather need to use a group of people. How much of the original material should be used for the judgement? The dataset covers several years of material and the printouts consist of hundreds of

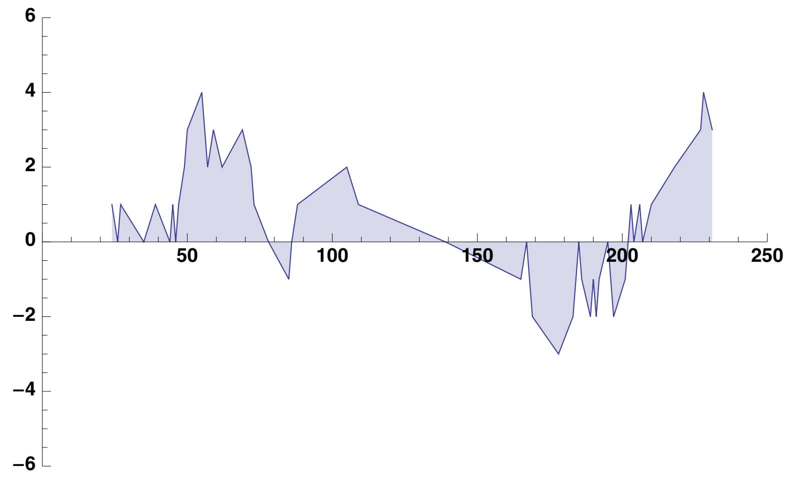


Fig. 7. Fashion. The groom balance for this subject fluctutates around zero. The person gives approximately as much as she gives.

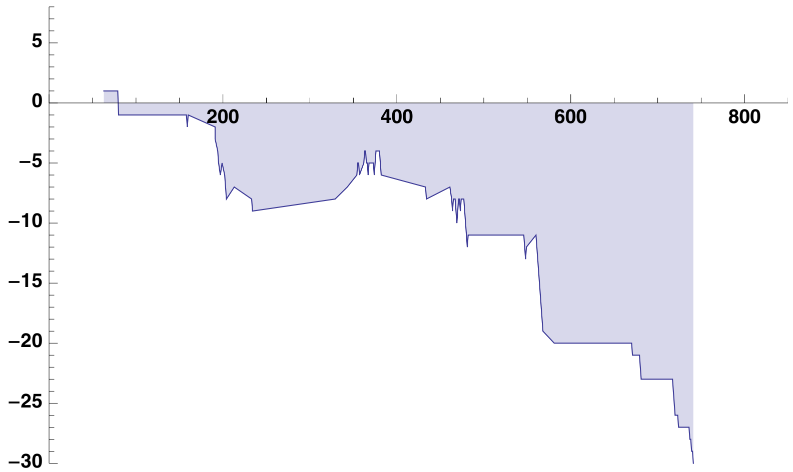


Fig. 8. Science. The groom balance for this subject stays negative. The person gives much more than he gets.

pages. It will be difficult to recruite persons to read that many posts. This will especially be true for the Science group, where the posts were very long and the content often was very scholarly. After considering these questions the following method was chosen.

Method of Validation. 40 people were recruited to the evaluation group, 17 males and 23 females. The mean age was 22 years (range 19-30). Each person read the printouts of 80 posts from the Fashion group. Each person read a different

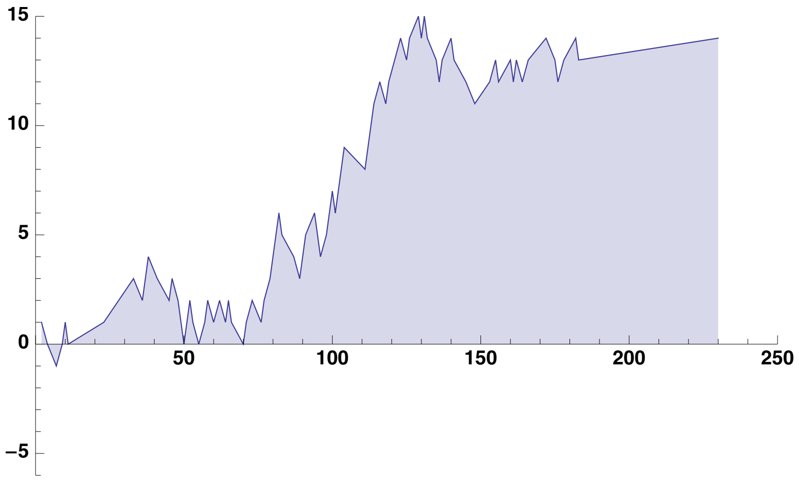


Fig. 9. Fashion. The groom balance for this subject gets more positive. The person gets much more than she gives.

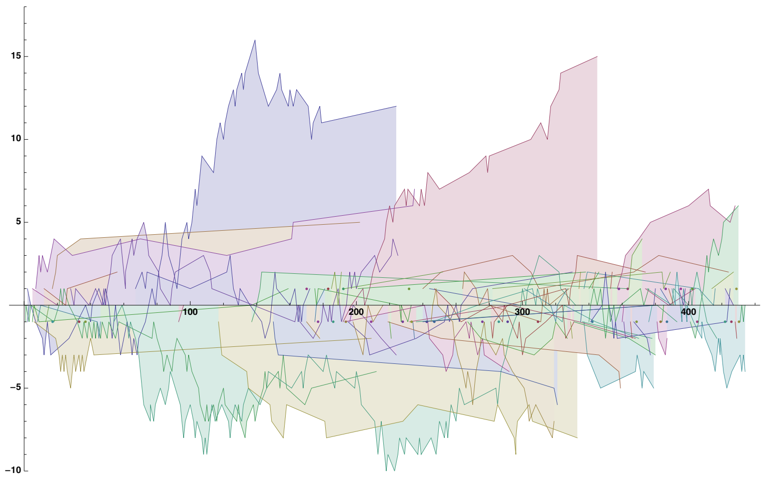


Fig. 10. Fashion. The combined groom balance for the whole group. Individuals with high positive groom balance replace each other during the time period studied.

set of posts, but the set of posts were overlapping so that one read posts 1-80 and another read posts 40-120 and so on. Thus every post was read by two different persons. They had 30 minutes to do the reading. After reading the posts the persons were asked to give their subjective impression of the different group members status in the group. The instruction was phrased as “pick out the participants in the discussion that you think have the highest status”. To do this they were presented with a response sheet including a list of the names of all group members that had appeared in the set of posts that they had read.

They were asked to give status points in the following way: the subject with the highest status was assigned 10 points, the subject with the next highest status was assigned 8 points, no 3 gets 6 points, no 4 gets 5 points, no 5 gets 3 points and finally no 6 get 2 points. The rest get no points at all. The status points were added up so that every group member had a total amount of status points. The ranking based on this value, was used for the validation.

Intra Validator Correlation. Since each group member had been judged by at least two different evaluators, it was possible to split the data set into two sets of data based on different evaluators to assess the intra evaluator correlation. A total of 2 x 598 judgments were given in the process of validation. When the correlation of these two sets of judgments was calculated it was 0.39. When *the sum* of all ranking points for all subjects was calculated for the two sets (each including 187 sums of points) the correlation between the two lists were 0.97. This means that the individual of status varied between individual evaluators but that the sum of points for each individual poster was very similar in the two calculations based on different judgments.

Results of the Validation. The ranking based on the collective subjective judgments of the evaluator group is called the evaluator ranking.

The correlation between the groom based ranking and the evaluator ranking was 0.85.

If the groom based method was used used to pick out the five people with highest status in the group the correspondence was 100%. Similarly for picking out the ten best it was a correspondence of 90% and for the best half of the group (94/187) it was 76%.

7 Discussion

In this study a number of similarities was found between the two groups. The total number of posts made by different individuals in the group followed a power-law like distribution. This was similar for the two groups. We could also see signs of scale-free self-similarity in the sense that if the data was broken up in four parts, each representing a certain time-period, the distribution of posts seemed to follow a power-law in all of the four periods. This is in accordance with the observations of Wilkinson who was able to estimate the power law exponent to between 2,35 to 1,5 [10]. (The data in this study suggests an exponent of 0,5). Other possibilities for explanations of the lognormal distribution has been proposed, like novelty decay [11] and changes in how easily content is seen by users [12].

The distribution of grooms between the group members seemed to follow a power law just as the distribution of posts. The number of grooms was extremely unevenly distributed. In both groups there was an inner circle of 8-10 people that received about 50% of the grooms. In the center of the inner circle was a core of a 4-5 people that together received 39-40% of the total number of grooms. In both groups the individuals in the inner core were from the majority gender of the group and, mostly, from English-speaking countries.

Another similarity was that in both groups the probability of making a second post was higher if a person was groomed after their first post compared to if they were not groomed after their first post. Also, in both groups there seemed to be reciprocity in the giving of grooms. The probability of beeing groomed was higher if a person had groomed someone else compared to if a person had not groomed someone else.

The introduction of the concept of groom as a directed social action made it possible to construct grooming networks for the two groups. The networks clearly showed the tight social connections between the people in the inner circles, and the dominance of the inner core. The community modularities for the two groups were similar: 0,55 for the Fashion group and 0,44 for the Science group.

Another similarity between the two groups was the distribution of posts over language categories. English speaking subjects made more than twice as many posts per person as the people from non-English speaking countries. In both groups people from non-English speaking countries groomed less but were more rewarded for their grooms when they did, compared with people from English speaking countries.

Gender related differences were similar but mirror imaged in the two groups. The under-represented gender made around 2 posts per person compared to 6-10 per person for the majority gender group. The gender also influenced the probability of getting groomed in a similar way. The under-represented gender was less likely to be groomed after their first post compared to people of the majority gender.

In both of the groups we could see that the choice of nickname seemed to affect the probability of getting groomed. Choosing a name was better than choosing a word, and choosing an non-pronouncable letter combination was the worst.

The analysis also enabled us to identify some differences between the groups. In the accumulated post graph we could in fact see that the Fashion group was dying out as indicated by the decreasing slope of the curve at the end of the observed time period. The Science group however showed no such slowing down, but instead the activity level kept constant.

The graph over group member activity over time uncovered that there were differences between the groups in terms of turn-over of the set of active subjects. In the Fashion group no subject remained active the whole time, in fact we could see a complete replacement of the active members so that the people active in the last quarter of the time period was a completely different set of people than the set that was active in the first quarter. In the Science group however, there was a set of subjects that remained steadily active for the whole time period studied.

Finally, the concept of groom enabled us to construct the groom balances. By doing so we could identify different types of grooming behavior. Most individuals showed a groom balance fluctuating around zero that indicates that they give about as many grooms as they receive. A small group of individuals get much more than they give and thus show a steadily rising groom balance. This may be the signature of a high-status person. Some individuals gave a lot of grooms but consistently stayed on a negative groom balance. This may correspondingly be the signature of a low-status person. The groom balance for the whole Fashion group showed that when a person with a very high groom balance left the group, another persons groom balance started to rise so that there always was at least one person with the signature of a high-status individual.

A clear difference between the groups showed up in the groom balances for the whole groups. In the Science group there was one individual who had written a book. This book was frequently cited in the discussion and because of that this group members groom balance sky-rocketed out of range. There was also another individual in the group who was paid as a facilitator. As a consequence of his role he answered very many questions from newcomers. Politely, responding with dear John Doe, he thus gave away many more grooms than he got. So even if he got quite a few grooms himself, his groom balance dived consistently the whole time period. The groom balance thus enabled us to identify that such special individuals were present in the group.

Discussion of the Method of Grooming Analysis. The mentions of names could not be automatically identified in 4% of the grooms for the Fashion group. The most common problem was that a name was shortened so that a part of the name, most often the last part was omitted. A few of the cases were due to misspellings, for instance replacing a vowel with another vowel as in writing *Sephie* instead of *Sophie*. Solving this problem require more elaborate string pattern recognition than the ones used in this program.

The grooming analysis can only be used to rank order the persons in the group that have received grooms. The ones that have not received any grooms at all however can not be rank ordered by this method as they all will have a total number of grooms equal to zero.

8 Conclusion

This study show that we can get a new perspective on the social dynamics in groups by introducing the concept of groom and applying the method of grooming analysis. When applied to the observed online discussion groups it gave additional insights into the behavior of the group members.

Based on the performed analyses the behavior of the discussion groups can now be described as follows. There is a constant inflow of people who *check out* the discussion groups. Some of these people are interested enough to make a post. If they get groomed, the probability increases that they will make a second post. If not, they will most likely leave the group. In this way a small subset of the whole group is filtered out that will keep posting. Among this active group there is an inner-circle that grooms each other. In the center of this group there is an inner core of a few people that receive a large amount of grooming.

This development is self-similar in the sense that it doesn't matter which time period we look at, there is always this center core of a few people that gets about 40% of the grooms given in that period. The center core may consist of different people in the different time periods.

Groups evolve to be gender specific. This effect is due to two reasons, first the two groups attracted different interest from females and males respectively, secondly that the under-represented gender got less return on investment for their given grooms, this in turn affected the probability of posting and thus the probability of getting groomed and so on in a self-reinforcing way that filters them out from the group.

The number of posts and the number of grooms are related in a complex way. The more you post the more you get groomed. But the reverse is also true, the more you get groomed the more you post. This reward some people and punish others so that we get this concentration of the social interaction to an inner core of a few people while the majority of people are in fact left out of the social interactions and have very low activity in the group. The validation showed that using the total number of grooms to rank order the group members in terms of social status resulted in a ranking with a correlation of 0.85 to that obtained with 40 independent subjective evaluators. This is a strong indicator that the grooming analysis actually reveal something "real" about the social interactions in the group.

In situations where there are no available data about quantity or quality of user contributions, or to data about links between group members, the method of grooming analysis can be used as an alternative to assess status. To generalize the methods to larger groups, further work needs to be done, in particularly the method of automatically identify names must be improved

References

1. Coie, J.D., Kupersmidt, J.B.: A Behavioral Analysis of Emerging Social Status in Boys Groups. Child Development 54(6), 1400– – 1416 (1983)
2. Mazur, A.: Biosocial Model of Status in Face-to-face Primate Goups. Social Forces 64(2) (1985)
3. Maisonneuve, N.: Basic Social Attention Analysis. The webblog of INSEADs Centre for Advanced Learning Technologies, CALT (2007), `http://www.slideshare.net/n.maisonneuve/social-analysis?from=fblanding`
4. Yan, M., Vassileva, J., Grassman, W.: A System Dynamics Approach to Study Virtual Communities. In: The 40th Hawaii International Conference on System Science. IEEE Press, New York (2007)
5. Zhang, Y.W., Tanniru, M.: An Agent-based approach to Study Virtual Learning Communities. In: The 35th Hawaii International Conference on System Science. IEEE Press, New York (2002)
6. Agarwal, N., Liu, H., Tang, L.: Identifying the influential bloggers in a community. In: The International Conference on Web Search and Web Data Mining, Palo Alto, California, USA (2008)

7. Sabater, J., Sierra, C., Ueda, N.: 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 - AAMAS 2002, AAMAS 2002 (2002)
8. Simkin, M.V., Roychowdhury, MV.P.: A mathematical theory of citing. Journal of the American Society for Information Science and Technology 58(11), 1661–1673 (2007)
9. Joyce, E., Kraut, R.E.: Predicting Continued Participation in Newsgroups. Journal of Computer-Mediated Communication (3), 723–747 (2006)
10. Wilkinson, D.M.: Strong regularities in online peer production. In: The 9th Conference on Electronic Commerce 2008, Chicago, Il, USA, pp. 302–309 (2008)
11. Wu, F., Huberman, B.: Novelty and collective attention. Proceedings of the National Academy of Sciences of the United States of America (45), 17599–17601 (2007)
12. Lerman, K., Rey, M.: Social Information Processing in Social News Aggregation Anatomy of Digg. Information Sciences (2), 1–17 (2007)

Exploring Gender Differences in Member Profiles of an Online Dating Site Across 35 Countries

Slava Kisilevich and Mark Last

Department of Computer and Information Science
University of Konstanz
slaks@dbvis.inf.uni-konstanz.de
http://www.informatik.uni-konstanz.de/arbeitsgruppen/infovis/mitglieder/slava-kisilevich/
Department of Information System Engineering
Ben-Gurion University of the Negev
mlast@bgu.ac.il
http://www.bgu.ac.il/~mlast/

Abstract. Online communities such as forums, general purpose social networking and dating sites, have rapidly become one of the important data sources for analysis of human behavior fostering research in different scientific domains such as computer science, psychology, anthropology, and social science. The key component of most of the online communities and Social Networking Sites (SNS) in particular, is the user profile, which plays a role of a self-advertisement in the aggregated form. While some scientists investigate privacy implications of information disclosure, others test or generate social and behavioral hypotheses based on the information provided by users in their profiles or by interviewing members of these SNS. In this paper, we apply a number of analytical procedures on a large-scale SNS dataset of 10 million public profiles with more than 40 different attributes from one of the largest dating sites in the Russian segment of the Internet to explore similarities and differences in patterns of self-disclosure. Particularly, we build gender classification models for the residents of the 35 most active countries, and investigate differences between genders within and across countries. The results show that while Russian language and culture are unifying factors for people's interaction on the dating site, the patterns of self-disclosure are different across countries. Some geographically close countries exhibit higher similarity between patterns of self-disclosure which was also confirmed by studies on cross-cultural differences and personality traits. To the best of our knowledge, this is the first attempt to conduct a large-scale analysis of SNS profiles, emphasize gender differences on a country level, investigate patterns of self-disclosure and to provide exact rules that characterize genders within and across countries.

Keywords: Social Networking Sites, Self-disclosure, Gender differences, Classification trees, Multidimensional Scaling, Hierarchical Clustering.

1 Introduction

Rapid technological development of the Internet in recent years and its worldwide availability has changed the way people communicate with each other. Social Networking Sites such as Facebook or MySpace gained huge popularity worldwide, having hundreds of millions of registered users. A major reason for the increased popularity is based on social interaction, e.g. networking with friends, establishing new friendships, creation of virtual communities of mutual interests, sharing ideas, open discussions, collaboration with others on different topics or even playing games. The key component of all SNSs is the user profile, in which the person cannot only post personal data, e.g. name, gender, age, address, but also has the opportunity to display other aspects of life, such as personal interests, (hobbies, music, movies, books), political views, and intimate information. Photos and videos are equally important for a self-description. All SNSs allow the user to upload at least one photo. Most mainstream SNSs also feature video uploading.

Various research communities have realized the potential of analysis of the SNS-phenomenon and its implication on society from different perspectives such as law [1], privacy [2–4], social interaction and theories [5–9]. Many hypotheses and social theories (gender and age differences, self-disclosure and self-presentation) have been raised and tested by social scientists using the context of Social Networks. Statistical analysis is the widely used instrument for analysis among social scientists and rely on the sampling rather than on data collected from an entire population segment.

The common approach to perform Social Network analysis is to analyze a sample of available user profiles or to conduct a survey using convenience samples (e.g. students in a particular university) by presenting descriptive statistics of the sample data and performing significance tests between dependent variables [2, 3, 8, 10]. The major drawback of such approach with respect to Social Networks is that in light of the large population of SNSs, which can vary from tens to hundreds of millions of users, living in all parts of the world, the results of the statistical analysis cannot be generalized for the whole population of users or a single nation or a single culture or genders, while theories can hardly be validated using only small samples. Moreover, Social Networks are heterogeneous systems in a sense that people may form closed sub-groups on different levels like country of living, national, or cultural with minimum interaction with other sub-groups and develop communication and self-presentational styles that are completely different from others due to cultural or national differences. For example, due to cultural differences, a theory of self-disclosure tested on students from American universities may be not be applicable to information obtained from students of Chinese universities, even if both groups use the same Social Networking Site. To the best of our knowledge, the state of the art Social Science research of Social Networks does not take into account the spatial or cultural components for the analysis of self-presentation differences (presumably due to the lack of sufficient data and difficulty involved in conducting cross-country studies).

Although, the problem and the importance of space and place in the Social Sciences was already highlighted a decade ago [11], this knowledge gap was not closed until to date. Therefore, in order to improve our understanding of social behavior, to analyze, to find hidden behavioral patterns not visible at smaller scales, and to build new theories of large heterogeneous social systems like Social Networks, other approaches and computational techniques should be applied [12].

In this paper, we answer the following hypothetical question: "Can we find some hidden behavioral patterns from user profiles in the large-scale SNS data beyond mere descriptive statistics?" We answer this question by applying a classification algorithm to the data obtained from more than 10 million profiles having more than 40 different attributes extracted from one of the largest dating sites in the Russian segment of the Internet. Specifically, we build gender classification models for most active countries and investigate what are particular differences between genders in one country and what are the differences in patterns of self-disclosure across countries. Self-disclosure can be defined as any information about himself/herself which a person communicates to others [13]. In the context of the current study, it refers to the information communicated by means of a person's online profile.

Dating sites can be considered as a special type of social networks where members are engaged in development of romantic relationship. Information revealed in the users' profiles is an important aspect for the assessment of potential communication, for maximizing the chances for online dating for the owner of the profile, and for minimizing the risks (e.g. misrepresentation) of online dating for the viewer of the profile. For this reason, in the broad context, assuming that the goal of the member of a dating site is to find a romantic partner, we investigate patterns of self-presentation that can vary from country to country and differ for both genders.

The preliminary results suggest that the classification model can successfully be used for analysis of gender differences between users of SNSs using information extracted from user profiles that usually contain tens of different categorical and numerical attributes.

To the best of our knowledge, this is the first attempt to conduct a large-scale analysis of SNS profiles for comparing gender differences on a country level using data mining approaches in the Social Science context.

2 Related Work

Gender differences have been studied long before the Internet became widely available. However, with the technological development of the Internet and proliferation of Social Networks, the research has focused on the analysis of online communities and differences between their members. Many studies were performed in the context of Internet use [14, 15], online relationships [5], ethnic identity [8], blogging [10], self-disclosure and privacy [2–4]. Though we could not find any related work on large-scale analysis of gender differences in social networks (except for [4]), we are going to review here some of the recent, mostly small-scale studies and findings about gender differences in social networking sites.

Information revelation, privacy issues and demographic differences between Facebook users were examined in [2] and [3]. [2] interviewed 294 students and obtained their profiles from Facebook. The goal of the survey was to assess the privacy attitudes, awareness of the members of the SNS to privacy issues, and the amount and type of information the users revealed in their profiles. It was found that there was no difference between males and females with respect to their privacy attitudes and the likelihood of providing certain information. Likewise, there was no difference between genders in information revelation. If some information is provided, it is likely to be complete and accurate. However, female students were less likely to provide their sexual orientation, personal address and cell phone number. [3] interviewed 77 students to investigate different behavioral aspects like information revelation, frequency of Facebook use, personal network size, privacy concerns and privacy protection strategies. Again, there were almost no differences between female and male respondents in the amount and type of the information revealed in their profiles. [4] analyzed about 30 million profiles from five social networks of Runet and conducted a survey among Russian speaking population to cross-check the finding extracted from the profiles and assess privacy concerns of members of Russian social networks. It was shown that there were differences between type of revealed information between females and males and these differences conditioned on the reported country of residence (20 most populous countries were presented). Particularly, males disclosed more intimate information regardless of their country of origin. However, the country with the highest difference in the amount of disclosed intimate information was Russia (20.67%) and the lowest was Spain (5.59%). In addition, females from 17 countries revealed more information about having or not having children, economic and marital status, and religion. The only exceptions were females in Russia, Israel and England.

Social capital divide between teenagers and old people, and similarities in the use of the SNS were studied in [7] using profiles from MySpace social network. The results of the analysis indicate, among other criteria, that female teenagers are more involved in the online social interaction than male teenagers. Likewise, statistical tests showed that older women received more comments than older men. Additionally, linguistic analysis of user messages showed that females include more self-descriptive words in their profiles than males. Friendship connections, age and gender were analyzed in [6] using $15,043$ MySpace profiles. The results showed that female members had more friends and were more likely interested in friendship than males, but males were more likely to be interested in dating and serious relationships. In the study that analyzed emotions expressed in comments [9], it was found that females sent and received more emotional messages than males. However, no difference between genders was found with respect to negative emotions contained in messages.

Online dating communities are typically treated differently because goals of the dating sites are much more limited in terms of connection development and often bear intimate context, which for the most part shifts to the offline context. Issues such as honesty, deception, misrepresentation, credibility assessment, and credibility demonstration, are more important in the dating context than in the context of general purpose social networks. Researchers are particularly interested in the analysis of self-presentation and self-disclosure strategies of the members of dating sites for achieving their goal to successfully find a romantic partner. The authors of [5] interviewed 349 members of a large dating site to investigate their goals on the site, how they construct their profiles, what type of information they disclose, how they assess credibility of others and how they form new relationships. The study found that cues presented in the users' profiles were very important for establishing connections. These cues included very well-written profiles, lack of spelling errors and uploaded photos. The last time the user was online considered to be one of the factors of reliability. Most of the respondents reported that they provided accurate information about themselves in the profiles.

3 Data

The data used in this paper was collected from one of the largest dating sites in Runet: *Mamba*[1]. According to the site's own statistics (June 3, 2010), there are $13,198,277$ million registered users and searchable $8,078,130$ profiles. The main features of the service is the user profile and search option that allows searching for people by country, gender, age and other relevant attributes. The friend list is discrete, so other registered users cannot know with whom a user is chatting. The friend list is implicitly created when the user receives a message from another user. There are no means to block unwanted users before they send a message. However, users get a *real* status by sending a free SMS to the service provider and confirming his/her mobile phone number. This allows the users with the *real* status to communicate with and get messages only from the real people. The user may exclude his/her profile from being searchable, but most of the profiles are searchable and accessible to unregistered users.

The user profile consists of six sections, where each section can be activated or deactivated by the user. Table 1 shows the names of sections and attribute parameters available in every section. We excluded the *About me* section, in which the user can describe himself in an open form, some intimate attributes of the *Sexual preference* section and the option to add multimedia (photos or videos). The attributes are divided into two categories. In the first category, only one value can be selected for the attribute (denoted as "yes" in the Single selection column), other attributes contain multiple selections (denotes as "no" in the Single selection column). Most of the attributes also contain an additional free text field that allows the user to provide his/her own answer. If the user decides not to fill in some field, the attribute won't be visible in his/her profile. The user

[1] `http://www.mamba.ru/`

can extend his/her main profile by filling two surveys. The one survey is provided by MonAmour site[2], owned by Mamba and contains about 100 different questions that estimate the psychological type of the respondent according to four components scaled from 0 to 100: *Spontaneity, Flexibility, Sociability, Emotions.* Another survey is internal and contains 40 open questions like *Education, Favorite Musician, etc*[3].

In order to collect the data, we developed a two-pass crawler written in C#. In the first pass the crawler repeatedly scans all searchable users which results in a collection of a basic information about the user such as *user id, profile URL, number of photos in the profile, and country and city of residence.* In the second pass, the crawler downloads the user's profile, checks if it is not blocked by the service provider and extracts all the relevant information, which is described in Table 1.

Within a two-month period, between March and June 2010, we extracted information from 13,187,295 millions users, where 1,948,656 million profiles were blocked, leaving us with 11,238,639 million valid profiles.

4 Methodology

In this section we describe the data mining process that includes data selection, data transformation and model construction.

4.1 Data Selection

The data preparation and selection is very crucial for the data mining process. If sampled data is not a representative of the whole dataset, the data mining process will fail to discover the real patterns. Another aspect of data preparation is related to user profiles. As was already discussed in Sections 1 and 2, the ultimate goal of members of the dating site is to find a romantic partner. Since this kind of activity may involve elements of intimacy, persons employ different strategies to balance the desire to reveal information about themselves and stay anonymous (for example, the profile without a photo). Moreover, many people may run several user profiles for different purposes.

In order to minimize the impact of fake profiles (e.g. empty profiles or profiles containing the minimal amount of information) on the pattern mining, we employed a four level filtering process. First, the profiles of persons who filled the external survey on the MonAmour site (described in Section 3) were retrieved. Since the respondent should answer about 100 questions, it is unlikely that the person has non-serious intentions on the dating site. Second, we retrieved profiles who filled additional external survey that includes about 40 questions. Next, the users with the status "real" were retrieved and finally, the users who uploaded

[2] `http://www.monamour.ru/`

[3] At the time of writing this paper, the structure of the profile and some of the fields were changed by the service provider.

Table 1. Profile sections and attributes

Section	Attributes	# of options	Single selection	Possible choice
Personal	Age	-	yes	20
	Gender	2	yes	Male/Female
	Zodiac	12	yes	Capricorn...
Acquaintances	Looking for	5	no	Man/Woman/Man+Woman/ Man+Man/Women+Woman
	Partner's age	8	no	16-20/21-25/26-30/31-35/ 36-40/41-50/51-60/61-80
	Aim	5	no	Friendship/Love/Sex/ Marriage/Other
	Marriage	4	yes	Married/Live separately Sham marriage/No
	Material support	3	yes	Want to find a sponsor/ Ready to become a sponsor/ No sponsor is required
	Kids	4	yes	No/I'd like to have/ Live together/Live separately
Type	Weight	1	yes	70 kg.
	Height	1	yes	180 cm.
	Figure	8	yes	Skinny/Regular/Sportive...
	Body Has	2	no	Tattoo, Piercing
	Hair on the head	7	yes	Dark/Grey-haired...
	Smoking	4	yes	No/Yes/Seldom/Drop
	Alcohol	3	yes	No/Yes/Seldom
	Drugs	8	yes	No/Yes/Drop/Dropped
	Profession	-	-	Open field
	Economic conditions	4	yes	Occasional earnings/ Stable and small income/ Stable and average income/ Wealthy
	Dwelling	6	yes	No steady place/Apartments/Dorm/ Live with Parents/Friend/Spouse
	Languages	87	no	English/German...
	Day regimen	2	yes	Night owl/Lark
	Life priorities	8	no	Carrier/Wealth/Family/Harmony/ Sex/Self-realization/ Public activity/Other
	Religion	7	no	Christianity/Atheism/Other...
Sexual preferences	Orientation	3	yes	Hetero/Homosexual/Bi
	Heterosexual experience	4	yes	Yes/No/Little/Other
	Excitement	18	no	Smell/Latex/Tattoos/Piercing...
	Frequency	6	yes	At least once a day/Other Several times per Day/Week/Month Not interested in sex
Interests	Leisure	14	no	Reading/Sport/Party...
	Interests	19	no	Science/Cars/Business...
	Sports	12	no	Fitness/Diving...
	Music	11	no	Rock/Rap...
Other	Car	76	yes	Nissan...
	Mobile Phone	50	yes	Ericsson...

at least one photo and no more than one hundred photos were extracted. Table 2 shows the demographic statistics by country and gender. It also shows how many profiles were selected for mining and the resulted percentage of females and males in the selected instances. The selected age range was from 18 to 73.

Table 2. Demographic statistics of the 35 most active countries and statistics related to the sampled data

Country	**Total**	**Females %**	**Males %**	**# instances**	**Sampled Females %**	**Sampled Males %**
Russia	7,844,969	65	35	3,039,762	45	55
Ukraine	1,257,890	52	48	711,586	49	51
Kazakhstan	456,940	57	43	201,775	46	54
Belarus	310,819	45	55	217,143	47	53
Germany	128,168	43	57	79,140	41	59
Azerbaijan	102,726	31	69	44,150	15	85
Uzbekistan	86,010	22	78	40,485	25	75
Moldova	78,835	40	60	54,561	44	56
Armenia	68,334	43	57	22,382	18	82
Georgia	67,554	20	80	33,022	21	79
Latvia	53,433	59	41	29,512	53	47
Estonia	48,243	52	48	26,731	48	52
USA	47,111	40	60	30,517	41	59
Israel	42,627	37	63	27,296	37	63
England	35,938	62	38	14,989	35	65
Turkey	35,001	16	84	23,884	14	86
Lithuania	34,795	59	41	16,481	48	52
Kyrgyzstan	32,798	36	64	16,592	38	62
Italy	18,389	42	58	13,635	43	57
Spain	18,220	38	62	11,503	40	60
France	11,988	36	64	7,187	33	67
Turkmenistan	11,609	31	69	5,952	34	66
Canada	10,623	36	64	6,604	35	65
Greece	10,092	30	70	7,088	30	70
Tajikistan	9,879	14	86	3,917	14	86
Czech	9,401	42	58	6,443	43	57
Poland	9,376	65	35	3,171	36	64
Finland	7,186	41	59	4,460	40	60
Sweden	6,348	32	68	4,045	28	72
Norway	5,994	28	72	3,437	28	72
Belgium	5,849	34	66	3,102	29	71
Bulgaria	5,649	31	69	3,719	28	72
Ireland	5,603	35	65	4,061	37	63
Austria	5,474	36	64	3,065	35	65
China	5,277	34	66	3,453	41	59

4.2 Data Transformation

Almost all the attributes described in Table 1 were selected for inclusion into the model (except for *Weight*, *Height*, and *Mobile Phone*). Numerical attributes such as age, number of photos and number of words used in the "About me" section were discretized. The age was discretized into ten equal size bins. We analyzed the distribution of photos and words in "About me" section individually for females and males. Based on the data distribution, the number of photos was divided into three categories: *none* if no photo was uploaded by the user, *normal* if the number of photos was between 1 and 8 for females and between 1 and 6 for males, *high* if the number of photos was between 9 and 16 for females and between 7 and 10 for males, *very high* if the number of photos was larger than 16 for females and larger than 10 for males. The number of words used in the "About me" section was divided into three categories: *none* if nothing was written, *normal* if the number of words was between 1 and 24 for females and between 1 and 22 for males, *high* if the number of words was between 25 and 260 for females and between 23 and 243 for males, *very high* if the number of photos was larger than 260 for females and larger than 243 for males.

Attributes such as *Car, Languages, Religion, Leisure, Interests, Sports, Music* and attributes describing body characteristics whose exact values are not important for classification but only the fact of their disclosure in a profile, were encoded as binary attributes: if the information about any of these attributes was revealed, it was encoded as *True*, otherwise it was treated as *False*. On the other hand, attributes, whose values are used for classification were encoded as multi-valued categorical attributes. For example, the *Marriage* attribute has four explicit options and one implicit *no answer*. In this case the four options were encoded like *1,2,3,4*, and *0* in the case of non-disclosure. Another group of attributes that may take more than one value (when the user chooses more than one answer) was decomposed into separate binary attributes representing distinct categories. For example, the user can select any of the 5 different categories related to the aim on the site (*Aim* attribute). In case a person selects some category, a binary *True* is assigned to that attribute, otherwise *False* is assigned (*Aim* not disclosed). Two binary attributes that were composed from the *Looking for*, namely *Looking for a man* and *Looking for a woman* were removed since they are found in the majority of profiles, highly correlated with the opposite gender and trivial in terms of gender classification.

4.3 Model Construction

Our research hypothesis is that specific gender differences exist on the country level as well as there are differences between the same-genders in different countries. The differences should be expressed in specificity of attributes and values that describe the gender. In other words, we hypothesize that profiles of females and males living in the same country have unique characteristics, which characterize the gender of the owner of the profile. In addition, we hypothesize that, although the main characteristic of the users of the featured dating site is

Russian language, cultural and national differences impact the characteristics of user profiles even for people of the same gender across countries. In our study, the data mining process that can capture unique characteristics of the genders is based on decision tree learning, which constructs a classification model using input variables for prediction of the target class value (gender in our case).

We applied C4.5, a popular decision tree induction algorithm [16] to the sampled data for every country with the *gender* as a binary class attribute, using Weka data mining package [17].

Here is a general outline of the algorithm:

– Tree is constructed in a top-down recursive divide-and-conquer manner.
– At start, all the training examples are at the root.
– Attributes are categorical or continuous-valued.
– Examples are partitioned recursively based on selected attributes.
– Split attributes are selected on the basis of a heuristic or statistical measure (in our experiments, we have used information gain).
– The complete tree can be post-pruned to avoid overfitting.

A decision tree can be easily converted into a set of classification rules (one rule per each terminal node). Tables 5 and 6 show examples of classification rules extracted from the induced decision trees. We left all the options in the default state namely: the minimum number of instances per leaf was 2, pruned decision tree, 0.25 pruning confidence factor. Table 3 shows the total number of classification rules and the number of rules by gender generated for every country.

5 Analysis

The purpose of this section is to analyze the data and the model described in Section 4. We apply a number of analytical steps to test our hypotheses that there are differences between genders and that these differences are country-dependent.
The analytical steps are:

(1) Analysis of the sampled data
(2) Analysis of the quantity of rules that classify females and males
(3) Cross-country similarity
(4) Gender characterization

5.1 Data Analysis

As was mentioned in Section 2, we applied four filtering steps to minimize the effect of false profiles. By inspecting the initial and resulting number of females and males (Table 2), we can deduce cross-country differences on the gender level.

Russia, Poland, England, Latvia, Lithuania, Kazakhstan, Ukraine, and Estonia are countries in which the number of female users outnumber male users. The difference is as large as 30% for Russia and Poland and as small as 4% in Ukraine

Table 3. The total number of rules by country and the number of generated rules by gender

Country	All Rules	Female	Male
Russia	70,719	34,605	36,114
Ukraine	21,181	10,315	10,866
Kazakhstan	5,863	2,815	3,048
Belarus	8,343	4,062	4,281
Germany	3,221	1,482	1,739
Azerbaijan	781	338	443
Uzbekistan	945	418	527
Moldova	1,754	818	936
Armenia	490	191	299
Georgia	649	267	382
Latvia	1,721	812	909
Estonia	1,433	692	741
USA	1,581	723	858
Israel	1,024	453	571
England	784	350	434
Turkey	350	149	201
Lithuania	1,150	534	616
Kyrgyzstan	656	292	364
Italy	699	327	372
Spain	791	382	409
France	483	209	274
Turkmenistan	234	106	128
Canada	404	175	229
Greece	334	156	178
Tajikistan	134	48	86
Czech	587	280	307
Poland	256	127	129
Finland	314	149	165
Sweden	307	135	172
Norway	193	91	102
Belgium	225	94	131
Bulgaria	158	79	79
Ireland	267	124	143
Austria	227	106	121
China	226	104	122

and Estonia. After applying the four filtering steps, only Latvia remains a single country among the eight mentioned above where the number of females still outnumber male users, however, the difference decreases from 18% to 6%. Since the number of people that do not have photos in their profile is much larger than the number of people who do not meet the requirement of the first three filtering steps, we may conclude that more females do not have photos in their profiles.

As the photo is one of the important components of a dating site, we can also assume that male users apply more efforts to find romantic partner than females or that female users would likely establish a relationship independent of physical appearances.

Uzbekistan, Georgia, Turkey and Tajikistan are the four countries that stand out in the difference between the number of male and female users: Uzbekistan (56%), Georgia (60%), Turkey (68%), Tajikistan (72%). This number does not almost change after applying the filtering step. Armenia, Poland, Russia, Tajikistan, England, Azerbaijan, Kazakhstan, Uzbekistan, Lithuania, and Georgia lose more than 50% of users, while Greece, Ireland, and Italy lose less than 30% after applying the filtering process.

5.2 Model Analysis

We use several different metrics to analyze gender differences in homogeneity and heterogeneity as well as variability of information revealed in user profiles by analyzing classification rules. The metrics, presented in Table 4, include the average amount of male/female members per rule (larger numbers indicate higher homogeneity), the number of male/female rules that cover 90% of the instances in the sampled dataset (larger numbers indicate higher heterogeneity), and the ratio of the number of male/female rules to the entire male/female population (larger numbers indicate higher variability). We also compute the difference between the male and the female rule ratios.

The inspection of the average number of female and male users that are classified per one rule (Table 4), shows that there are only three countries Latvia (4%), Lithuania (1%), and Ukraine (1%), in which the average number of females classified per rule is larger than in the other 32 countries. Such countries as Tajikistan (28%), Uzbekistan (33%), Armenia (40%), Georgia (42%), Azerbaijan (65%), and Turkey (80%) are countries with the largest difference between the average number of female and male users classified per rule out of 32 countries where the average number of males per rule outnumber females. However, in 31 countries the number of rules that cover 90% of the population is larger for females with the greatest difference in Azerbaijan (71%), Bulgaria (66%), Turkey (66%), Uzbekistan (65%), Georgia (65%), Poland (53%), Greece (51%), while Finland (4%), Estonia (9%), Lithuania (10%), Latvia (29%) are the only four countries where the number of rules that cover 90% of the population is larger for males. This finding may suggest that female users are more creative in profile construction and provide more heterogeneous information about themselves, while males reveal more homogeneous information to describe themselves. This is also supported if we inspect the amount of rules generated for females and males relative to the number of females and males in the data set (Table 4). The amount of rules in the percentage relationship is higher in 32 cases for females. The highest relative amount of rules for female population is in Sweden (10.92%), Poland (11.13%), Belgium (10.45%), Czech Republic (10.11%)

and the lowest in Ukraine (2.96%) and Russia (2.53%). For the male population, the highest relative amount of rules is observed in Czech Rupublic (8.36%), Lithuania (7.19%), Latvia (6.55%), and the lowest in Azerbaijan (1.18%) and Turkey (0.98%).

Another interesting observation are cross-country and cross-gender variabilities of the relative amount of rules. The difference between the highest (Sweden) and the lowest (Russia) relative amount of rules for females is 9.39%, while the difference between the highest (Czech Republic) and the lowest (Turkey) relative amount of rules for males is 7.38%. If we assume that information disclosed in the users' profile is a deliberate and considerate act that also reflects personal traits of a person (otherwise the profile would have been randomly filled) and the variability of rules shows the variability in different facets of personal traits then our observation of cross-country variability between females and males in relative amount of rules is orthogonal to previous studies. For example, [18] showed that the considerable gender differences in personality traits are among European and American cultures, whereas the miniscule differences are among African and Asian cultures. In our case, the highest cross-gender difference in the relative amount of rules is in Tajikistan (6.20%), which is an Asian country, Sweden (6.01%), and Norway (5.33%) while the lowest is in Estonia (0.06%) and Ukraine (0.04%). Other Asian countries such as Azerbaijan (3.92%) or Turkey (3.48) are ranked on the seventh and tenth place, respectively among the countries with highest variability of the 35 countries we analyzed. Consequently, our results indicate that the gender differences are not emphasized by the Russian-speaking users in masculine countries [19]. The Masculine Index of scandinavian countries is very low according to Hofstede Masculine Index [20], while Sweden and Norway share the second and the third places in the magnitude in differences between females and males. However, these differences may be attributed to the fact that the Swedish and Norwegian Russian-speaking members of the dating site have a stronger influence of their original culture rather than the culture of their current residence.

Any decision tree construction algorithm builds rules by determining the best attributes that build up the tree. The attribute at the root of the tree is the first attribute selected and, thus, is the best in the classification model to discriminate between genders. Inspection of the root attributes of the models reveals four groups of countries:

(1) The majority of countries (14 in total) namely Spain, Kyrgyzstan, Lithuania, Italy, Ireland, Greece, Estonia, England, China, Moldova, Latvia, Kazakhstan, Israel, and Belarus are characterized by the attribute *AimSex* (the aim on the site is to find a partner for having sex).
(2) Turkmenistan, Poland, Norway, France, Czech Republic, Canada, Bulgaria, Austria, USA, Uzbekistan, Ukraine, and Russia are countries in which the classification tree is splitted according to the *Car* attribute.
(3) Turkey, Tajikistan, Georgia, Armenia, and Azerbaijan are characterized by *MinMaxAge*. This attribute holds the desired age range of a romantic partner.

(4) The remaining four countries: Sweden, Finland, Belgium, and Germany are characterized by the *kids* attribute, which specifies whether the person does or does not have kids, whether the kids live in family or separately or if the person wants to have kids.

Table 4. The average amount of females and males classified per rule, the number of rules that cover 90% of the instances in the sampled dataset, percentage of rules relative to the female and male population, and difference between relative amount of rules

Country	**Females Per Rule**	**Males Per Rule**	**90% Rule Coverage Male**	**90% Rule Coverage Female**	**Female Rel. Amount Rules (%)**	**Male Rel. Amount Rules (%)**	**Difference**
Russia	40	46	5,707	3,815	2.53	2.16	0.37
Ukraine	34	33	2,060	1,951	2.96	2.99	-0.04
Kazakhstan	33	36	590	513	3.03	2.80	0.24
Belarus	25	27	1,094	1,006	3.98	3.72	0.26
Germany	22	27	498	442	4.57	3.72	0.84
Azerbaijan	20	85	179	52	5.10	1.18	3.92
Uzbekistan	24	58	165	57	4.13	1.74	2.39
Moldova	29	33	245	193	3.41	3.06	0.34
Armenia	21	61	93	47	4.74	1.63	3.11
Georgia	26	68	121	42	3.85	1.46	2.39
Latvia	19	15	267	374	5.19	6.55	-1.36
Estonia	19	19	235	259	5.39	5.33	0.06
USA	17	21	310	261	5.78	4.77	1.01
Israel	22	30	155	125	4.49	3.32	1.16
England	15	22	170	108	6.67	4.45	2.22
Turkey	22	102	73	25	4.46	0.98	3.48
Lithuania	15	14	224	251	6.75	7.19	-0.44
Kyrgyzstan	22	28	116	91	4.63	3.54	1.09
Italy	18	21	143	112	5.58	4.79	0.79
Spain	12	17	204	135	8.30	5.93	2.38
France	11	18	115	97	8.81	5.69	3.12
Turkmenistan	19	31	52	29	5.24	3.26	1.98
Canada	13	19	87	75	7.57	5.33	2.24
Greece	14	28	88	43	7.34	3.59	3.75
Tajikistan	11	39	32	19	8.75	2.55	6.20
Czech	10	12	157	138	10.11	8.36	1.75
Poland	9	16	75	35	11.13	6.36	4.77
Finland	12	16	68	71	8.35	6.17	2.19
Sweden	8	17	79	69	11.92	5.91	6.01
Norway	11	24	57	34	9.46	4.12	5.33
Belgium	10	17	56	48	10.45	5.95	4.50
Bulgaria	13	34	44	15	7.59	2.95	4.64
Ireland	12	18	66	46	8.25	5.59	2.66
Austria	10	16	63	50	9.88	6.07	3.81
China	14	17	52	42	7.35	5.99	1.36

5.3 Cross-Country Similarity

In Section 4.3 we applied a decision tree construction process to the user profiles from 35 countries, and generated models that contain a number of rules that discriminate between females and males in a specific country. As mentioned already, classification trees are used for predicting the target class value. Usually, in order to estimate a classifier's predictive performance, the model is evaluated on a separate test set. In the context of our analysis, we have applied the classification rules generated for each country to the data of other 34 countries. The high classification rate in this case should suggest that there is a high similarity between user profiles (including user information disclosure) across countries. As a result of the evaluation, we have a 35-dimensional vector of classification accuracies for each of the 35 countries (including the training accuracy of the model on the data that was used to induce the model). We applied Multidimensional Scaling (MDS)[21], a widely used data exploratory technique, on the 35×35 matrix of classification accuracies. MDS performs transformation of multidimensional space into a two-dimensional coordinates by preserving the relative distances (we used squared Euclidean distance measure) between original multi-dimensional vectors. Thus, the countries located close to each other on the two-dimensional graph are more similar in information disclosure between their residents than countries that are located farther away.

Figure 1 shows the results of multidimensional scaling. It is possible to visually discern eight clusters according to the similarity in user profiles. Russia and Ukraine are located close to each other and can form the first cluster. Germany, US, Kazakhstan, Belarus, and Moldova are located close to each other and form the second cluster. Uzbekistan, Israel, Kyrgyzstan, Greece, England, Spain, and France are members of the third cluster. The fourth cluster includes Armenia, Turkey, Azerbaijan, Georgia, Turkmenistan, Belgium, and Canada. Sweden, Austria, Ireland, Finland and Czech Republic are in the fifth cluster. Italy is located equally distant from other countries and can be a single country in the sixth cluster. Estonia, Lithuania, and Latvia are in the seventh cluster. Tajikistan, China, Poland, Norway, and Bulgaria are located farther than other countries. We assign them to an eighth cluster. The first and the seventh cluster include countries that are located geographically close to each other. This may suggest that cultural similarities between those countries play a crucial role in the similarity of user profiles. Similar observations were reported in [22, 23] in the study of personality traits. Other clusters include a mix of close and faraway countries. For example, the fourth cluster contains five Asian countries geographically close to each other such as Armenia, Azerbaijan, Turkmenistan, Turkey, and Georgia as well as two countries situated in Europe and America. A notable feature of the cluster two is that Kazakhstan and Germany are located close to each other. While those countries are not located close geographically, it is known that a significant number of Russian Germans now living in Germany, immigrated from Kazakhstan during 1990s where their ancestors had lived in the late 19th Century.

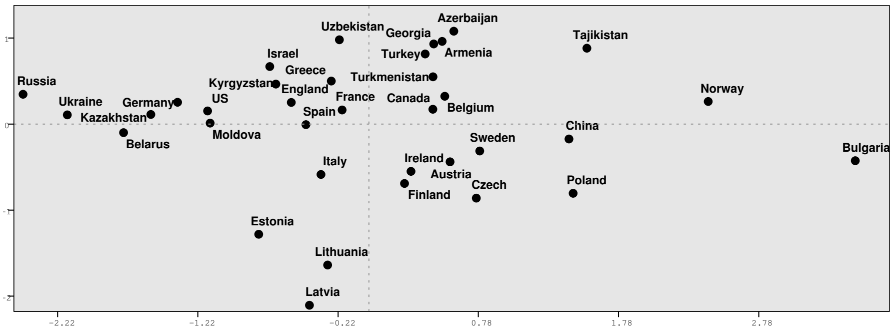


Fig. 1. MDS plot of similarity of information disclosure across 35 countries

A more consistent way to summarize the similarities is to explicitly cluster the countries according to the similarities and differences in member profiles. We applied Farthest Neighbor (complete linkage) hierarchical clustering using cosine similarity between 35-dimensional vectors. Results of the clustering are shown in Figure 2. It can be clearly seen that some clusters are discerned by geographically close countries. For example, Armenia, Georgia, Turkey, Azerbaijan, and Tajikistan are linked together at the first level of hierarchical clustering. Likewise, Russia, Ukraine, and Belarus or Uzbekistan and Turkmenistan. On the highest level of clustering, Estonia, Lithuania, and Latvia are linked with all other countries suggesting the considerable difference in member profiles between the three countries and the rest of the countries.

5.4 Gender Characterization

Since the space limitation does not allow us to present the whole list of rules generated for every gender and country, we provide a number of rule examples picked from the set of most frequent rules. We arbitrarily selected two representative countries from every visible cluster produced by the multidimensional scaling (Figure 1) described in the previous section. Tables 5 and 6 show frequent classification rules that distinguish between females and males across countries. The rightmost column shows the number of males (M) and females (F) that correspond to each rule.

The inspection of the rules show some clear-cut patterns. The sex and car components are dominated in “male” rules (rules that classify a person as a male). Information about kids and the desired age of a romantic partner is dominated in “female” rules. A noteworthy feature found in “male” rules is that they are relatively short. There are cases where a single attribute can classify a person as a male like in the case of Bulgaria and China.

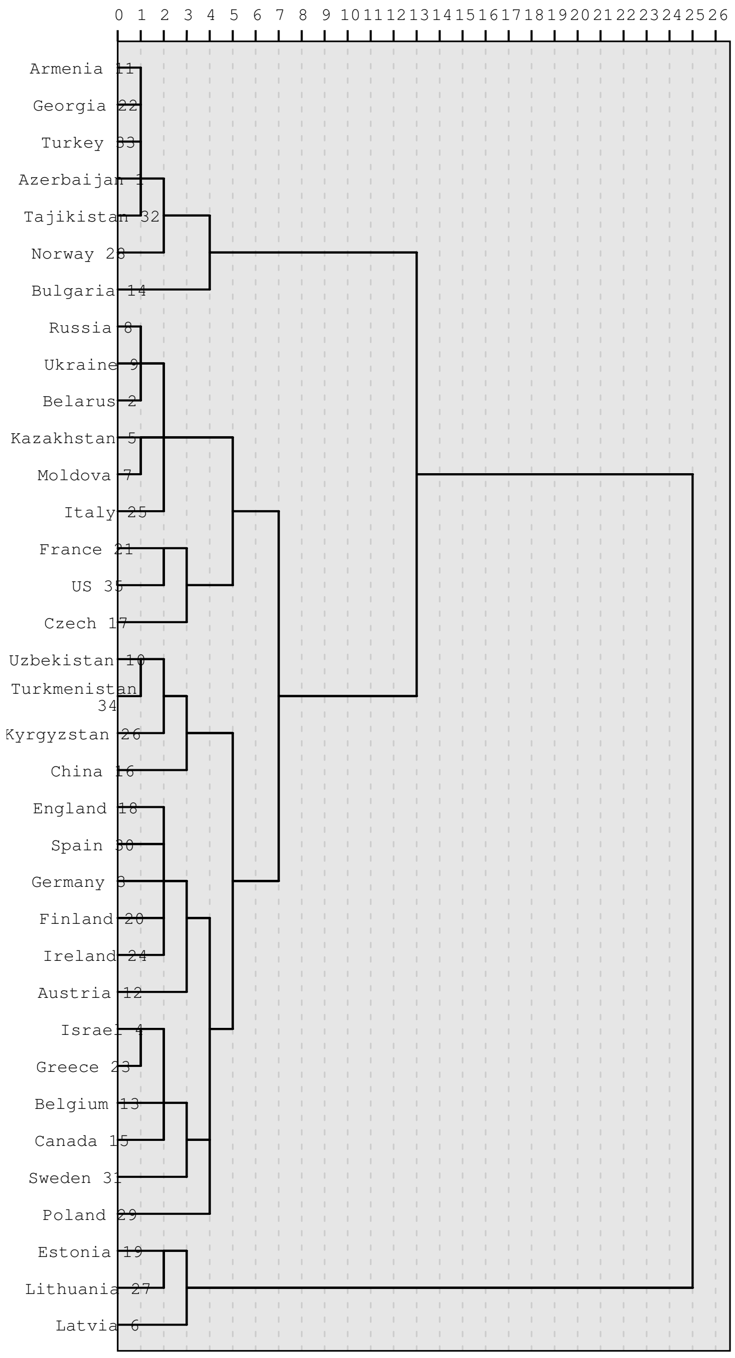


Fig. 2. Hierarchical clustering diagram of similarities of information disclosure accross 35 countries

Table 5. Frequent rules that classify females across countries. The rightmost column shows the number of females (F) in the dataset that are classified by the corresponding rule and the number of males (M) that correspond to the same rule.

Country	**Rule**	Gender (F/M)
Russia	IF I have kids AND I am not married AND I specified what I'd do on my free day AND information about a car and smoking habits is not revealed AND I did not reveal that my aim is to have sex	F (21,712/160)
Ukraine	IF I have kids AND I am not married AND my age is between 23.5 and 29 AND information about a car is not revealed AND I did not reveal that my aim is to have sex	F (7,965/103)
Germany	IF I have kids AND I am not married AND I did not reveal that my aim is to have sex AND information about a car is not revealed	F (4,786/262)
USA	IF I have a car AND I do not have any photo AND I revealed information about my body AND I wrote nothing about myself AND I did not reveal that my life priority is sex AND information about my profession is not revealed	F (322/20)
Israel	IF I have kids AND I am not married AND I did not reveal that my aim is to have sex	F (2,853/223)
England	IF I am looking for a person between 41 and 60 AND information about a car is not revealed AND I did not reveal that my aim is to have sex	F (102/5)
Belgium	IF I have kids AND I have between 1 and 8 photos AND my life priority is to have a family AND I did not reveal that my aim is to have sex AND I am not looking for a friendship with a female or female couple	F (133/22)
Turkey	IF I am looking for a person between 31 and 40 AND my age is between 29 and 34.5 AND information about other aims is not revealed	F (104/6)
Czech	IF I revealed some of my life priorities but did not select that my life priority is self-realization AND I am looking for a person between 18 and 20 AND I revealed what I like in sex AND I earn well AND information about a car or how frequent I'd like to have sex were not revealed	F (276/1)
Sweden	IF I have kids AND information about a car or how frequent I'd like to have sex is not revealed AND I did not reveal that my aim is to have sex	F (242/31)
Italy	IF I am looking for a person between 18 and 20 AND harmony is my life priority AND I do not take drugs AND I did not reveal that my aim or life priority is sex AND I did not reveal my orientation, heterosexual experience or information about a car	F (330/64)
Lithuania	IF I have kids AND I am not married AND I did not reveal that my aim is to have sex AND information about a car is not revealed	F (1,001/59)
Latvia	IF I have kids AND information about a car is not revealed AND I did not reveal that my aim is to have sex	F (3,667/339)
Bulgaria	IF I wrote between 1 and 24 words about myself AND my hobby is music AND information about a car is not is not revealed AND I did not reveal that my aim it to have sex or how frequent I'd like to have sex	F (242/70)
China	IF harmony is my life priority AND I reveal some information about my body AND information about a car or economic condition is not revealed AND I did not reveal that my aim is to have sex	F (375/103)

Table 6. Frequent rules that classify males across countries. The rightmost column shows the number of males (M) in the dataset that are classified by the corresponding rule and the number of females (F) that correspond to the same rule.

Country	**Rule**	Gender (M/F)
Russia	IF I have a car AND I am a heterosexual AND I am looking for a person between 18 and 30 AND I am not looking for a friendship with a male and female couple	M (15,973/431)
Ukraine	IF I have a car AND I am a heterosexual AND my aim is to have sex AND I am not looking for a friendship with a male and female couple	M (18,461/902)
Germany	IF I do not have kids AND my aim is to have sex AND information about a car is not revealed AND I am a heterosexual	M (951/101)
USA	IF I have a car AND I have between 1 and 6 photos AND I am looking for a person between 18 and 20 AND my life priority is sex	M (688/66)
Israel	IF my aim is sex AND information about a sexual orientation is not revealed	M (986/140)
England	IF my aim is sex AND I am a heterosexual AND I revealed information about my interests	M (667/35)
Belgium	IF I do not have kids AND I have a car	M (345/42)
Turkey	IF I have a car AND I am looking for a person between 18 and 20	M (3807/152)
Czech Republic	IF I revealed how frequent I'd like to have sex AND information about a car is not revealed	M (344/74)
Sweden	IF I do not have kids AND my aim is to have sex	M (189/15)
Italy	IF I have a car AND I do not have kids AND my aim is marriage AND I did not reveal that my aim is to have sex	M (911/113)
Lithuania	IF I have a car AND I am looking for a person between 18 and 20 AND information about kids is not revealed AND I did not reveal that my aim is to have sex	M (1022/241)
Latvia	IF my aim is to have sex AND I am a heterosexual AND I wrote nothing about myself AND I am not looking for a friendship with a homosexual male, and heterosexual couples	M (748/61)
Bulgaria	IF I have a car	M (1156/132)
China	IF I have a car	M (245/30)

6 Conclusions

In this paper we investigated gender differences and patterns of information dislosure between countries in the context of dating sites using the Data Mining approach.

We applied decision tree construction algorithm to the user profiles from 35 most active countries using more than 10 million profiles from one of the biggest dating sites in the Russian segment of the Internet. We analyzed the induced classification rules and outlined differences between genders within and across countries. We used Multidimensional scaling and hierarchical clustering to analyze similarities and differences in member profiles and information disclosure across countries. The Russian-speaking residents of some geographically close and culturally similar countries exhibit higher similarity in information disclosure between user populations living in those countries.

We showed that social phenomena can be investigated by applying data mining methods to large quantities of user profile data, and that statistical analysis alone is not enough for finding interesting patterns. Our research overcomes the limitations of most previous studies, where the analysis was performed on small, non-representative and non-generalizable samples of the user population. However, some uncertainty is associated with the large-scale analysis of real profiles mined from a social networking site, since the analyst cannot verify the real purpose of profile creation (whether it has a serious intention or was created for fun). At this point, we assume that the majority of SNS users have real profiles that reflect their real self. Automated cleaning of profile data may be a subject of future research.

Our study provided insights into the patterns of gender differences across countries. The reasons for such differences can be unlimited: influences of the hosting country' culture, immigration, spoken language, original culture, personal traits. Therefore, we could not provide exact explanations of such difference and did not attempt to speculate on possible reasons. Moreover, the meaning of gender differences could be explained by domain experts like anthropologists, culturalists, behaviorists or sociologists. Without a doubt further studies are necessary. Previous studies on gender differences [15, 24] have been carried out on a much smaller scale in the context of "digital divide"[4]. The results of such studies can affect design principles and guidelines and provide insights for the development of SNS and other information systems. However, these potential applications are beyond the scope of this paper.

The preliminary results provided in this paper are encouraging, though the work presented here is exploratory in nature. In our future work, we will apply more analytical methods to conduct all-embracing analysis of gender differences, user profiles, and information disclosure and work closely with social scientists to test hypotheses that so far have been evaluated on very limited amounts of user data.

[4] The phrase "digital divide" has been used to refer to a wide variety of inequities, including differential access to, contact with, and use of ICTs cross-nationally as well as between social and demographic groups within individual nations [15].

Acknowledgements. This work was partially funded by the German Research Society (DFG) under grant GK-1042 (Research Training Group “Explorative Analysis and Visualization of Large Information Spaces”)

References

1. Nelson, S., Simek, J., Foltin, J.: The Legal Implications of Social Networking. Regent University Law Review 22(1), 2 (2009)
2. Acquisti, A., Gross, R.: Imagined communities: Awareness, information sharing, and privacy on the facebook. In: Danezis, G., Golle, P. (eds.) PET 2006. LNCS, vol. 4258, pp. 36–58. Springer, Heidelberg (2006)
3. Young, A., Quan-Haase, A.: Information revelation and internet privacy concerns on social network sites: a case study of facebook. In: Proceedings of the Fourth International Conference on Communities and Technologies, pp. 265–274. ACM, New York (2009)
4. Kisilevich, S., Mansmann, F.: Analysis of privacy in online social networks of Runet. In: Proceedings of the 3rd International Conference on Security of Information and Networks. ACM, New York (2010)
5. Ellison, N., Heino, R., Gibbs, J.: Managing impressions online: Self-presentation processes in the online dating environment. Journal of Computer-Mediated Communication 11(2), 415 (2006)
6. Thelwall, M.: Social networks, gender, and friending: An analysis of MySpace member profiles. Journal of the American Society for Information Science and Technology 59(8), 1321–1330 (2008)
7. Pfeil, U., Arjan, R., Zaphiris, P.: Age differences in online social networking-A study of user profiles and the social capital divide among teenagers and older users in MySpace. Computers in Human Behavior 25(3), 643–654 (2009)
8. Grasmuck, S., Martin, J., Zhao, S.: Ethno-Racial Identity Displays on Facebook. Journal of Computer-Mediated Communication 15(1), 158–188 (2009)
9. Thelwall, M., Wilkinson, D., Uppal, S.: Data mining emotion in social network communication: Gender differences in MySpace. Journal of the American Society for Information Science and Technology (2009)
10. Pedersen, S., Macafee, C.: Gender differences in British blogging. Journal of Computer-Mediated Communication 12(4), 1472 (2007)
11. Goodchild, M., Anselin, L., Appelbaum, R., Harthorn, B.: Toward spatially integrated social science. International Regional Science Review 23(2), 139 (2000)
12. Kleinberg, J.: The convergence of social and technological networks. Commun. ACM 51(11), 66–72 (2008)
13. Cozby, P.C.: Self-disclosure: A literature review. Psychological Bulletin 79(2), 73 (1973)
14. Golub, Y., Baillie, M., Brown, M.: Gender Differences in Internt Use and Online Relationships. American Journal of Psychological Research 3(1) (2007)
15. Jones, S., Johnson-Yale, C., Millermaier, S., Pérez, F.: US College Students’ Internet Use: Race, Gender and Digital Divides. Journal of Computer-Mediated Communication 14(2), 244–264 (2009)
16. Quinlan, J.: C4. 5: programs for machine learning. Morgan Kaufmann, San Francisco (1993)
17. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.: The WEKA data mining software: An update. ACM SIGKDD Explorations Newsletter 11(1), 10–18 (2009)

18. Costa, P., Terracciano, A., McCrae, R.: Gender differences in personality traits across cultures: Robust and surprising findings. Personality and Social Psychology 81(2), 322–331 (2001)
19. Hofstede, G.: Masculinity and femininity: The taboo dimension of national cultures. Sage Publications, Thousand Oaks (1998)
20. Hofstede, G.: Culture's consequences: International differences in work-related values. Sage Publications, Thousand Oaks (1984)
21. Borg, I., Groenen, P.: Modern multidimensional scaling: Theory and applications. Springer, Heidelberg (2005)
22. Allik, J., McCrae, R.: Toward a geography of personality traits: Patterns of profiles across 36 cultures. Journal of Cross Cultural Psychology 35(1), 13–28 (2004)
23. Schmitt, D., Allik, J., McCrae, R., Benet-Martinez, V., Alcalay, L., Ault, L., et al.: The geographic distribution of Big Five personality traits: Patterns and profiles of human self-description across 56 nations. Cross Cultural Psychology 38(2), 173 (2007)
24. Jackson, L.A., Zhao, Y., Kolenic III, A., Fitzgerald, H.E., Harold, R., Von Eye, A.: Race, gender, and information technology use: the new digital divide. CyberPsychology & Behavior 11(4), 437–442 (2008)

Community Assessment Using Evidence Networks

Folke Mitzlaff[1], Martin Atzmueller[1], Dominik Benz[1],
Andreas Hotho[2], and Gerd Stumme[1]

[1] University of Kassel, Knowledge and Data Engineering Group
Wilhelmshöher Allee 73, 34121 Kassel, Germany
[2] University of Wuerzburg, Data Mining and Information Retrieval Group
Am Hubland, 97074 Wuerzburg, Germany
{mitzlaff,atzmueller,benz,stumme}@cs.uni-kassel.de,
hotho@informatik.uni-wuerzburg.de

Abstract. Community mining is a prominent approach for identifying (user) communities in social and ubiquitous contexts. While there are a variety of methods for community mining and detection, the effective evaluation and validation of the mined communities is usually non-trivial. Often there is no evaluation data at hand in order to validate the discovered groups.

This paper proposes an approach for (relative) community assessment. We introduce a set of so-called *evidence networks* which are capturing typical interactions in social network applications. Thus, we are able to apply a rich set of implicit information for the evaluation of communities. The presented evaluation approach is based on the idea of reconstructing existing social structures for the assessment and evaluation of a given clustering. We analyze and compare the presented approach applying user data from the real-world social bookmarking application BibSonomy. The results indicate that the evidence networks reflect the relative rating of the explicit ones very well.

1 Introduction

With the rise of social applications, a wealth of data is stored, and finding relevant entries in the overwhelming user generated data repositories becomes more and more of a problem. Personalizing the access to such systems is a key approach for preventing users to get "lost in data". Promising approaches for such personalization are *user recommendation* or *community mining* techniques. Knowing a user's peer group is, for example, used to adjust search results according to his or her interests [17], or for showing the latest activities or most popular resources only for a set of relevant users.

Parallel to the rise of the Social Web, mobile phones became more and more powerful and are equipped with more and more sensors, giving rise to Mobile Web applications. Today, we observe the amalgamation of these two trends, leading to a Ubiquitous Web, whose applications will support us in many aspects of the daily life at any time and any place. Data are now available which were never accessible before. We expect therefore that the approach presented in this paper will be extendable to ubiquitous applications especially to sensor networks as well.

However, ultimately judging whether users are related or not is rather difficult since any given pair of users shares some properties. Hence, it is usually non-trivial to objectively assess the quality of a given community. As a consequence, Siersdorfer proposed an evaluation paradigm which is based on the notion of reconstructing *an existing social structure* [31], for example, considering friendship links in social applications. Transferring this paradigm to the evaluation of community mining techniques, we propose to assess a given community structure using *a set of existing social structures*. To this end, we introduce a set of so-called *evidence networks* which are capturing typical interactions in social network applications. These interactions can be regarded as proxies for social relations between users, i. e., as implicit connections. Thus, evidence networks provide *evidences* of social relations, but do not require explicit interaction and linking.

Our framework allows to compare the community structure that is computed with a given community mining algorithm with these evidence networks. It provides thus a method for evaluating and comparing different community mining approaches. For the assessment, we apply standard community quality functions, e. g., modularity [26] and conductance [10].

Existing "social structures" are often sparse (compared to the large number of evaluation objects present in clustering and community detection approaches) and usually not publicly available. In addition to explicit networks (e. g., by adding someone as a "friend"), this work also analyzes relations which are *implicitly* acquired in a typical "Web 2.0" application (e. g., by visiting a user's profile page). Our hypothesis in using these networks, which we call evidence networks, is the assumption, that the set of social interactions is drawn from a certain "social population", thus the interactions indicate connections in this distribution, and they manifest themselves with varying degree in different networks. By considering samples of such a "social constellation", we aim to collect evidences for the underlying user relatedness.

Considering implicit evidence networks for evaluation encompasses several advantages. In every application where users may interact, there are implicit evidence networks, even if no explicit user relationship is being implemented. Implicit networks may also be captured anonymously on a client network's proxy server. Typically implicit networks are also significantly larger than explicit networks. Similar interaction networks accrue in the context of ubiquitous applications (e. g., users which are using a given service at the same place and time). Unfortunately no dataset containing such interaction was available during the evaluation, but these interactions lead to implicit user relationships which naturally fit into the framework of evidence networks described in this section.

This paper proposes an approach for the evaluation of communities using implicit information formalized in evidence networks. Our context is given by social applications such as social networking, social bookmarking, and social resource sharing systems. The proposed evaluation paradigm is based on the notion of *reconstructing existing social structures*: This paradigm suggests to measure the quality of a given division of the users by assessing the corresponding community structure in an existing social structure: We basically project the different clusters according to the division of users on an existing network, and assess the created structures using measures for community evaluation. The contribution of this paper is *not* the presentation of a new algorithm for detecting communities. Instead, we rather focus on a better understanding of what a good user community is and how to assess and evaluate a given community allocation:

- We propose evidence networks for community assessment and evaluation. These evidence networks are thoroughly analyzed with respect to the contained community structure.
- We apply standard community evaluation measures using the set of evidence networks. It is shown, that there is a strong common community structure across different evidence networks.
- The results suggest a basis for a new evaluation framework of community detection methods in social applications.

The context of the presented analysis is given by social applications such as social networking, social bookmarking, and social resource sharing systems, considering our own system BibSonomy[1] [5] as an example. But the presented analysis is not only relevant for the evaluation of community mining techniques, but also concerning their usage for building new community detection or user recommendation algorithms, among others.

The rest of the paper is structured as follows: Section 2 first describes basic notions of the presented approach. Then, it describes this task in detail and introduces our novel approach together with the concept of evidence networks and their characteristics. After that, we analyze and compare in Section 4 the features of the networks using data from the real-world BibSonomy system: We define different emerging networks of user relatedness, and analyze these in detail applying our proposed method. Finally, Section 5 concludes the paper with a summary and interesting directions for future work.

2 Evidence Networks for Community Evaluation

In the following, we briefly introduce basic notions, terms and measures used in this paper. For more details, we refer to standard literature, e. g., [13]. After that, we describe and define several explicit and implicit networks for the evaluation of communities. Finally, we discuss related work.

2.1 Preliminaries

This section summarizes basic notions and terms with respect to graphs, explicit and implicit relations, communities, and community measures.

A *graph* $G = (V, E)$ is an ordered pair, consisting of a finite set V which consists of the *vertices* or *nodes*, and a set E of *edges*, which are two element subsets of V. A *directed graph* is defined accordingly: E denotes a subset of $V \times V$. For simplicity, we write $(u, v) \in E$ in both cases for an edge belonging to E and freely use the term *network* as a synonym for a graph. The *degree* of a node in a network measures the number of connections it has to other nodes. The *adjacency matrix* $A_{ij}, i = 1 \dots n, j = 1 \dots n$ of a set of nodes S with $n = |S|$ contained in a graph measures the number of connections of node $i \in S$ to node $j \in S$. A *path* $v_0 \rightarrow_G v_n$ of *length* n in a graph G is a sequence $v_0, \dots, v_n$ of nodes with $n \geq 1$ and $(v_i, v_{i+1}) \in E$ for $i = 0, \dots, n-1$. A *shortest path* between nodes u and v is a path $u \rightarrow_G v$ of minimal length. The *transitive closure* of a graph $G = (V, E)$ is given by $G^* = (V, E^*)$ with $(u, v) \in E^*$ iff there

[1] `http://www.bibsonomy.org/`

exists a path $u \rightarrow_G v$. A *strongly connected component (scc)* of G is a subset $U \subseteq V$, such that $u \rightarrow_{G^*} v$ exists for every $u, v \in U$. A *(weakly) connected component (wcc)* is defined accordingly, ignoring the direction of edges $(u, v) \in E$.

For a set V, we define a *relation* R as a subset $R \subseteq V \times V$. A relation R is naturally mapped to a corresponding graph $G_R := (V, R)$. We say that a relation R among individuals U is *explicit*, if $(u, v) \in R$ only holds, when at least one of u, v *deliberately* established a connection to the other (e. g., user u added user v as a friend in an online social network). We call R *implicit*, if $(u, v) \in R$ can be *derived* from other relations, e. g., it holds as a side effect of the actions taken by u and v in a social application. Explicit relations are thus given by explicit links, e. g., existing links between users. Implicit relations can be derived or constructed by analyzing secondary data.

The concept of a *community* is vague and can be intuitively defined as a group $\mathcal{C}$ of individuals out of a population $\mathcal{U}$ such that members of $\mathcal{C}$ are densely "related" one to each other but sparsely "related" to individuals in $\mathcal{U} \setminus \mathcal{C}$. In the following, a *community allocation* of a population $\mathcal{U}$ refers to a set of communities $\mathbb{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$ with $\bigcup_{1 \le i \le n} \mathcal{C}_i \subseteq \mathcal{U}$ and $\mathcal{C}_i \neq \emptyset$ for $1 \le i \le n$. Note that this also allows *overlapping communities*, i. e., $\mathcal{C}_i \cap \mathcal{C}_j \neq \emptyset$ may hold for some $i, j \in \{1, \ldots, n\}$.

This concept maps to vertex sets $C \subseteq V$ in graphs $G = (V, E)$ where nodes in C are densely connected but sparsely connected to nodes in $V \setminus C$. Though defined in terms of graph theory, the community concept remains vague. For a given graph $G = (V, E)$ and a community $C \subseteq V$ we set $n := |V|$, $m := |E|$, $n_C := |C|$, $m_C := |\{(u, v) \mid u, v \in C\}|$, $\overline{m}_C := |\{(u, v) \mid u \in C, v \notin C\}|$ and for a node $u \in V$ its degree is denoted by $d(u)$. Several approaches for formalizing communities in graphs exist and corresponding community structures were observed and analyzed in a variety of different networks [27,26,21,22].

In the context of evaluation measures for evidence networks we consider two measures: *Conductance* [22] and *Modularity* [26]. These consider the evaluation from two different perspectives. Modularity mainly focuses on the links *within* communities, while the conductance also takes the links between communities into account.

Conductance can be defined as the ratio between the number of edges within the community and the number of edges leaving the community. Thus, the conductance $C(S)$ of a set of nodes S is given by $C(S) = c_S/(2m_S + c_S)$ where c_S denotes the size of the edge boundary, $c_S := |\{(u, v) : u \in S, v \notin S\}|$ and m_S denotes the number of edges within S, $m_S := |\{(u, v) \in E : u, v \in S\}|$. More community-like partitions exhibit a low conductance, cf. [22]. The conductance of a set of clusters is then given by the average of the conductance of the single clusters.

The modularity function is based on comparing the number of edges within a community with the expected such number given a null-model (i. e., a randomized model). Thus, the modularity of a community clustering is defined to be the fraction of the edges that fall within the given clusters minus the expected such fraction if edges were distributed at random. This can be formalized as follows: The modularity $M(S)$ of a set of nodes S in graph G with its assigned adjacency matrix $A \in \mathbb{N}^{n \times n}$ is given by

$$M(A) = \frac{1}{2m} \sum_{i,j} (A_{i,j} - \frac{k_i k_j}{2m}) \delta(c_i, c_j) \, ,$$

where c_i is the cluster to which node i belongs, m denotes the number of edges in G and c_j is the cluster to which node j belongs; k_i and k_j denote i and j's degrees respectively; $\delta(c_i, c_j)$ is the *Kronecker delta* symbol that equals 1 iff $c_i = c_j$, and 0 otherwise. For *directed networks* the modularity becomes

$$M(A) = \frac{1}{m} \sum_{i,j} \big(A_{i,j} - \frac{k_i^{\text{out}} k_j^{\text{in}}}{m}\big) \delta(c_i, c_j)\,,$$

where k_i^{in} and k_j^{out} are i and j's in- and out- degree respectively [20].

The (Pearson) *correlation coefficient* r is used for measuring linear dependence between two random variables X and Y. We apply the sample correlation coefficient

$$r = \frac{\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n} (X_i - \bar{X})^2} \sqrt{\sum_{i=1}^{n} (Y_i - \bar{Y})^2}}.$$

were $\bar{X}$ and $\bar{Y}$ denote the sample mean of X and Y respectively [1].

2.2 Evidence Networks

Social networks and social resource sharing systems like BibSonomy usually capture links between users explicitly, e. g., in a *friend-network* or a *follower-network*. However, besides these explicit relations, there are a number of other *implicit* evidences of user relationships in typical social resource sharing systems. These are given by, e. g., clicklogs or page visit information. In some systems, it is also possible to copy content from other users. Then, the logging information can be transformed into a corresponding user-graph structure which we call *evidence network*, following [25].

In the following sections, we define typical explicit and implicit networks in the context of social bookmarking applications. All of these are implemented in the social resource sharing system BibSonomy, but are also found in other resource sharing and social applications. Even more implicit user interaction occurs in the context of ubiquitous web applications. Examples are users which are using a given service at the same place and time, or communication relationships based on proximity sensors [32], among many others. During our evaluation period we did not have access to such sensor data, but these interactions lead to implicit user relationships which naturally fit into the framework of evidence networks described in this section.

Explicit Relation Networks. In the context of the BibSonomy system, we distinguish the following explicit networks: The follower-graph, the friend-graph, and the group graph that are all established using explicit links between users. Formally, these graphs can be defined as follows:

- The *Follower-Graph* $G_1 = (V_1, E_1)$ is a directed graph with $(u, v) \in E_1$ iff user u follows the posts of user v, i. e., user u monitors the posts and is able to keep track of new posts of user v.
- The *Friend-Graph* $G_2 = (V_2, E_2)$ is a directed graph with $(u, v) \in E_2$ iff user u has added user v as a friend. In the BibSonomy system, the only purpose of the friend graph so far is to restrict access to selected posts so that only users classified as "friends" can observe them.

- The *Group-Graph* $G_3 = (V_3, E_3)$ is an undirected graph with $\{u, v\} \in E_3$ iff user u and v share a common group, e. g., defined by a special interest group.

Implicit Relation Networks. Concerning implicit relationships, we propose the following networks: The click-graph, the copy graph, and the visit graph that are built by analyzing the actions of users, i. e., clicking on links, copying resources, and visiting pages of other users, respectively. Formally, the graphs are defined as follows:

- The *Click-Graph* $G_4 = (V_4, E_4)$ is a directed graph with $(u, v) \in E_4$ iff user u has clicked on a link on the user page of user v.
- The *Copy-Graph* $G_5 = (V_5, E_5)$ is a directed graph with $(u, v) \in E_5$ iff user u has copied a resource, i. e., an publication reference from user v.
- The *Visit-Graph* $G_6 = (V_6, E_6)$ is a directed graph with $(u, v) \in E_6$ iff user u has navigated to the user page of user v.

Each implicit graph G_i, $i = 4, \ldots, 6$ is given a weighting function $c_i \colon E_i \to \mathbb{N}$ that counts the number of corresponding events (e. g., $c_5(u, v)$ counts the number of posts which user u has copied from v).

2.3 Evaluation Paradigm

Several approaches exist for assessing the quality of a given set of communities. Considering users as points in appropriate feature spaces, objective functions based on the resulting distribution of data points can be applied (e. g., overlaps of the user's tag clouds, [16]). Modeling inter-user relations in terms of graphs, various graph indices defined for measuring the quality of graph clusterings can be applied (see, e. g., [15] for a survey). These indices capture the intuition of internally densely connected clusters with sparse connections between the different clusters. Furthermore, the modularity measure (see Section 2.1) is based on the observation, that communities within social networks are internally more densely connected than one would expect in a corresponding null model, i. e., in a random graph.

Accordingly, most methods for community detection try to optimize the produced community division with respect to a given quality measure. However, care must be taken, since different measures might exhibit certain biases, i. e., they tend to reward communities with certain properties which might lead to respectively skewed community structures [22]. Given the diversity of user interests, no single quality measure can potentially reflect all reasons for two users being contained within the same or different communities (or even both). Ultimately, a user study can quantify, how well a given community structure coincides with the actual reception of the users.

Dealing with the related task of *user recommendations*, Siersdorfer [31] proposed an evaluation paradigm, which is based on the *reconstruction of existing social structures*. Applied to the community detection setting in the context of a social bookmarking system as BibSonomy, this paradigm suggests to measure the quality of a given division of the users by assessing the corresponding community structure in an existing social structure. For our evaluation paradigm we therefore transform this principle to evaluating community structures using evidence networks (see Section 2.2): Our input is given by an arbitrary community clustering of a given set of users – independent of any community detection method. This clustering is then assessed using the implicit evidence networks. We show in the evaluation setting that this procedure is consistent with applying explicit networks that contain explicit user links but are rather sparse compared to the evidence networks.

Concerning our application setting, BibSonomy incorporates three relations among users, all of which potentially can serve as a basis for such an evaluation, namely the *Friend-Graph*, the *Follower-Graph* and the *Group-Graph*. Before such a network can be utilized as a reference for quality assessments, it has to be thoroughly analyzed, since different structural properties may influence the resulting assessment, cf. [25]. But more importantly, one has to cope with the sparsity of the explicit user relations: The Friend-Graph of BibSonomy, for example, only spans around 1000 edges among 700 users of all 5600 considered users and all possible 30 million edges. Thus, feature spaces for users, for example, using tags or resources as describing elements potentially capture a richer set of relations than those modeled in the graphs. In the following, we therefore consider the much more dense *implicit evidence networks* as discussed in [25], which can be typically observed in a running resource sharing system. In our analysis, we investigate whether they are consistent with the existing explicit networks in BibSonomy as a reference for evaluating community detection methods.

3 Related Work

Despite the absence of well-established gold-standards, the growing need for automated user community assessment is reflected in a considerable number of proposed paradigms. Evaluation approaches of generated links between users can broadly be divided in content-based and structure-based methods (relying on given links between users). In the following, we discuss related work concerning community mining in general, community detection methods, evaluation measures, metrics and evaluation paradigms.

Fortunato [14] discusses various aspects connected to the concept of community structure in graphs. Basic definitions as well as existing and new methods for community detection are presented. This work is a good entry point for the topic of community mining. In [19], Lancichinetti presents a thorough comparison of many different state of the art community detection algorithms in graph. The performance of algorithms are compared relative to a class of adequately generated artificial benchmark graphs.

Karamolegkos et al. [16] introduced metrics for assessing user relatedness and community structure by means of the normalized size of user profile overlaps. They evaluate their metrics in a live setting, focussing on the optimization of the given metrics.

An LDA [7] based community detection method for folksonomies is presented in [17] which is evaluated indirectly by measuring the improvement of search results achieved by incorporating the mined community information. Using a metric which is purely based on the structure of graphs, Newman presents algorithms for finding communities and assessing community structure in graphs [28]. A thorough empirical analysis of the impact of different community mining algorithms and their corresponding objective function on the resulting community structures is presented in [22], which is based on the size resolved analysis of community structure in graphs [21].

Recently Siersdorfer et al. [31] proposed an evaluation technique for recommendation tasks in folksonomies which is based on the reconstruction of existing links (e. g., friendship lists). The performance of a given system is assessed by applying quality measures which are derived from established measures used in information retrieval. Schifanella et al. [30] investigated the relationship of topological closeness (in terms of the length of shortest paths) with respect to the semantic similarity between the users. Crandall et al. [12] discuss similarity and social influence in online communities, providing the general idea that friends interact similarly concerning activities of users. Their results indicate, that there are clear feedback effects between similarity between actors and future interactions.

Another aspect of our work is the analysis of implicit link structures which can be obtained in a running Web 2.0 system and how they relate to other existing link structures. Baeza-Yates et al. [4] propose to present query-logs as an implicit folksonomy where queries can be seen as tags associated to documents clicked by people making those queries. Based on this representation, the authors extracted semantic relations between queries from a query-click bipartite graph where nodes are queries and an edge between nodes exists when at least one equal URL has been clicked after submitting the query. Krause et al. [18] analyzed term-co-occurrence-networks in the logfiles of internet search systems. They showed that the exposed structure is similar to a folksonomy.

Analyzing Web 2.0 data by applying complex network theory goes back to the analysis of (samples from) the web graph [8]. Mislove et al. [24] applied methods from social network analysis as well as complex network theory and analyzed large scale crawls from prominent social networking sites. Some properties common to all considered social networks are worked out and contrasted to properties of the web graph. Newman analyzed many real life networks, summing up characteristics of social networks [29].

Concerning the approaches mentioned above, this work unifies topics of community mining, community evaluation, and social structures: We provide an approach for relative community assessment using the link structure of different (implicit) networks capturing user interactions. These so-called evidence networks are thoroughly analyzed with respect to the contained community structure. It is shown, that there is a strong common community structure across different evidence networks using standard community evaluation functions. The results suggest a basis for a new evaluation framework of community detection methods in social applications.

4 Evaluation

In the following, we first describe the data used for the evaluation of the evidence networks. After that, we describe the characteristics of the applied evidence networks, and discuss relations between the networks. After that, we present the conducted experiments. We conclude with a detailed discussion of the experimental results.

4.1 Evaluation Data and Setting

Our primary resource is an anonymized dump of all public bookmark and publication posts until January 27, 2010, from which we extracted *explicit* and *implicit* relations.

It consists of 175,521 tags, 5,579 users, 467,291 resources and 2,120,322 tag assignments. The dump also contains friendship relations modeled in BibSonomy concerning 700 users. Additionally, it contains the *follower* relation, which is explicitly established between user u and v, if u is interested in v's posts and wants to stay informed about new posts, as discussed above. Furthermore, we utilized the "*click log*" of BibSonomy, consisting of entries which are generated whenever a logged-in user clicked on a link in BibSonomy. A log entry contains the URL of the currently visited page together with the corresponding link target, the date and the user name[2]. For our experiments we considered all click log entries until January 25, 2010. Starting in October 9, 2008, this dataset consists of 1,788,867 click events. We finally considered all available apache web server log files, ranging from October 14, 2007 to January 25, 2010. The file consists of around 16 GB compressed log entries. We used all log entries available, ignoring the different time periods, as this is a typical scenario for real-world applications.

Table 1. High level statistics for all relations where U denotes the set of all users in BibSonomy

	Copy	Visit	Click	Follower	Friend	Group
$\|V_i\|$	1427	3381	1151	183	700	550
$\|E_i\|$	4144	8214	1718	171	1012	6693
$\|V_i\|/\|U\|$	0.25	0.58	0.20	0.03	0.12	0.10
#scc	1108	2599	963	175	515	90
largest scc	309	717	150	5	17	228
#wcc	37	11	55	37	140	89
largest wcc	1339	3359	1022	83	283	228

4.2 Characteristics of the Networks

In the following, we briefly summarize the link symmetry characteristics and degree distribution of the extracted networks and discuss its power-law distribution. The analysis is restricted to the large (weakly) connected components of the network.

Link symmetry: Mislove et al. [24] showed for Flickr, LiveJournal and YouTube that 60-80% of the direct friendship links between users are symmetric. Among others, one reason for this is that refusing a friendship request is considered impolite. However, the friendship relation of BibSonomy differs significantly. Only 43% of the friendship links between users are reciprocal.

When more features are available exclusively along friendship links (e. g., sending posts), the friendship graph's structure will probably change and links will get more and more reciprocal. But concerning the implicit networks we will see, that link asymmetry is determined by a structure common to all our implicit networks.

Degree distribution: One of the most crucial network properties is the probability distribution ruling the likelihood $p(k)$, that a node v has in- or out-degree k respectively. In most real life networks, the so called *degree distribution* follows a power law [11], that is $p(k) \sim k^{-\alpha}$ where $\alpha > 1$ is the exponent of the distribution. Online social networks [24], collaborative tagging systems [9], scientific collaboration networks [2] among others are shown to expose power law distributions.

[2] Note: For privacy reasons a user may deactivate this feature.

For comparability, we calculated a best fitting power law model for each distribution using a maximum likelihood estimator [11] and noted the corresponding Kolmogorov-Smirnov goodness-of-fit metrics in Table 2 for reference. All in- and out-degree distributions except those from the groups graph show a power law like behavior, though there are significant deviations.

Table 2. Power law parameters

	Copy	Visit	Click	Follower	Friend	Group
α_{in}	2.48	2.9	2.86	2.48	3.47	3.5
α_{out}	1.75	2.2	2.7	2.78	2.24	3.5
D_{in}	0.0603	0.0227	0.023	0.0278	0.0617	0.1503
D_{out}	0.0571	0.0364	0.0394	0.0919	0.0939	0.1503

4.3 Relations between Networks

Another basic evaluation considered associational properties between the networks, i. e., whether links present in one or more network are associated with links in another network. For the evaluation, we considered all possible links between users (user pairs) and assessed whether a link in a network existed, or not. Table 3 shows the associations for the friend graph. The associations are given in the form of subgroup rules (e. g., [3]), or association rules with the single element *friend*, denoting link membership in the friend graph, in the rule head. The parameters given in the table denote the support of the rule (i. e., the relative number of covered links), its confidence (or precision), the recall, and the F1-Measure, as the harmonic mean of precision and recall, e. g., [3]. Additionally, the table includes the *lift* of the rule denoting the relative increase in confidence/precision considering the default rule (with an empty precondition), i. e., is obtained by dividing the rule's confidence/precision by the default precision.

The first (baseline) rule therefore shows the default associations for the friend graph, i. e., the default confidence/precision for link membership is 5.5%. The second rule can be read as follows: IF users are connected in the *click* AND *copy* AND *visit* graphs, THEN the probability of being connected in the *friend* graph is 23.9% (confidence/precision), with a support of 1.7%, and an F1-Measure of 11.2%.

While the table shows significant increases of the rule confidences compared to the baseline measured by the lift parameter, the individual confidence values are rather low. This can be explained by the different sizes of the networks (the friend graph is the smallest network contained in the table). Additionally, the table shows interesting results comparing the individual networks: While the single *copy* network is only a limited predictor for friend-relationships (line 8) the combination with the *click* and/or *visit* networks significantly increases the associations quality, compared to considering the isolated networks (ll. 2, 3, 4, 6 and 7).

4.4 Applied Clustering Method

Starting our experiments we faced a vicious circle: For assessing the quality of a community structure, we need a preferably good method for obtaining such a structure in

Table 3. Associations between evidence networks and the friend graph. The table shows the support (SUP), confidence/precision (CONF), the recall (REC), the F1-Measure (F1M), and the Lift (LIFT) for the different associations, measured in percentages

#	RULE	SUP	CONF	REC	F1M	LIFT
1	friend $\leftarrow$	100	5.5	100	10.4	1.00
2	friend $\leftarrow$ click copy visit	1.7	23.9	7.3	11.2	4.35
3	friend $\leftarrow$ click copy	2	20.7	7.4	10.9	3.76
4	friend $\leftarrow$ copy visit	4.6	17.8	14.9	16.2	3.24
5	friend $\leftarrow$ click visit	7.7	13.5	19	15.8	2.45
6	friend $\leftarrow$ click	9.5	11.3	19.7	14.4	2.05
7	friend $\leftarrow$ visit	45.1	6.9	56.8	12.3	1.25
8	friend $\leftarrow$ copy	24.4	3.8	16.8	6.2	0.69

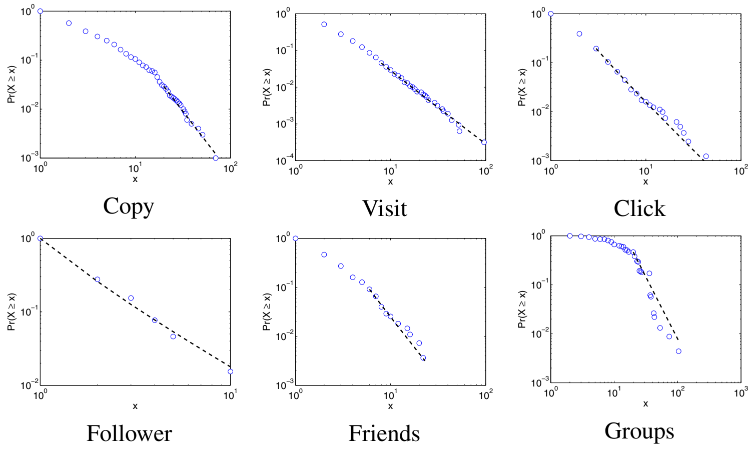


Fig. 1. In-degree distribution of the different evidence networks

the beginning. However, since we do not want to examine a particular clustering algorithm and prove its performance, we use a rather simple approach which is on the one hand easy to understand, on the other hand, it can be broadly parameterized and allows the construction of a randomized variety of initial clusterings.

First experiments were conducted using the well known *k-means algorithm* [23]. For that, each user u is represented by a vector $(u_1, \ldots, u_T) \in \mathbb{R}^T$ where T is the total number of tags and u_i is the total number of times user u assigned the tag i to resources in BibSonomy ($i = 1, \ldots, T$). The resulting clusters had poor quality, assigning most users to a single cluster. Due to the sparsity of the considered high dimensional vector space representation (there are more than $170,000$ tags), the underlying search for nearest neighbors fails (cf., e. g., [6] for a discussion).

To bypass this problem, we reduced the number of dimensions. There are a variety of approaches for dimensionality reduction. We chose to cluster the tags for building "*topics*", consisting of associated sets of tags. A user u is thus represented as a vector $\boldsymbol{u} \in \mathbb{R}^{T'}$ in the topic vector space, where $T' \ll T$ is the number of topics.

For our experiments, we used a *latent dirichlet allocation* [7] method for building topics, which efficiently build interpretable tag clusters and has been successfully applied in similar contexts to tagging systems (cf. [31]). In the following, our models are denoted with "*LDA-n-kMeans-k*", where n denotes the number of topics and k the number of clusters. In total, we obtained 40 different basic clusterings.

4.5 Experiments and Results

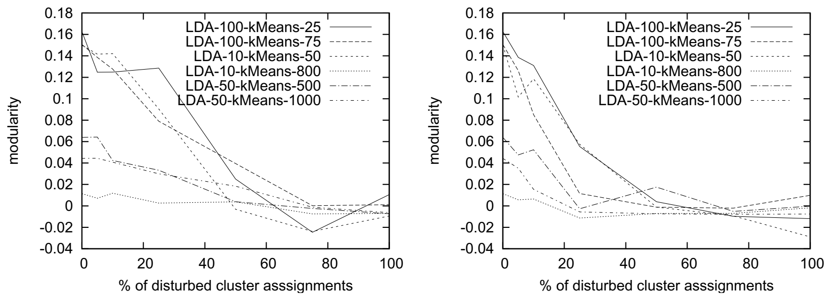
Our experiments aim at examining whether the *implicit evidence networks* described in Section 2.2 are admissible complements for the sparse *explicit networks*. This would justify using, e. g., the Visit-Graph and thus allow to assess more than 53$\%$ of the active users (in contrast to only 12$\%$ covered by the Friend-Graph) applying the evaluation paradigm "*reconstruction of existing social structures*" described in Section 2.3.

The most fundamental property of a sound measure is the relative discrimination of "better" and "worse" community structures, allowing algorithms to approximate optimal structures stepwise by applying local heuristics. For analyzing how quality assessment by applying the different evidence networks is sensitive to small disturbances, we conducted a series of randomized experiments.

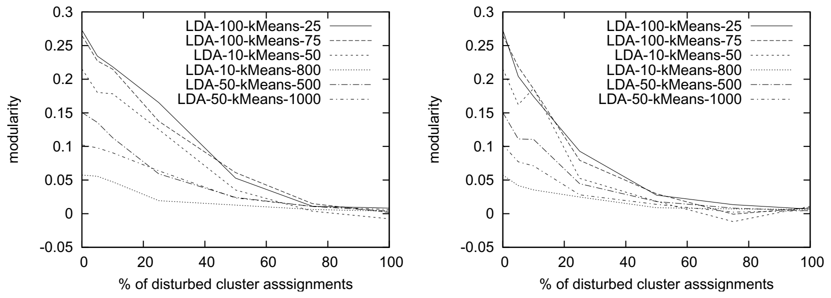
We started with community structures constructed by the basic feature clustering described above, using 10, 50, 100, and 500 topics, and constructing clusterings ranging from 10 to 1,000 clusters in total. Any clustering or community detection method could be used here (e. g., we also conducted the same series of experiments applying a graph clustering algorithm). We focussed on the applied method as it is easy to understand and can be broadly parameterized; it allows for a simple generation of a variety of (randomized) initial clusterings. We gradually added noise to these initial structures and at each step assessed the resulting community structure by calculating the quality measures described in Section 2.1 for the different evidence networks: Two different approaches for adding noise to a given division into communities were applied. The first approach (from now on called "Random" for short) randomly chooses a node u belonging to some community c_u. This node is than assigned to another randomly chosen community $c' \neq c_u$. Note that this kind of disturbance leads to a different distribution of cluster sizes. The second approach (from now on called "Shuffle") randomly swaps the community allocation of randomly chosen nodes belonging to different communities, which leads to community structures with the same community size distribution.

Figures 2 and 3 show the corresponding results of calculating the modularity for each evidence network at every level of disturbance in the underlying community structure (higher modularity values indicate stronger community structure). Similarly, Figures 4 and 5 show the results of calculating the conductance. For the ease of presentation, we selected from all considered clusterings a subset which represents a broad range of assessed community qualities. We emphasize that this experiment does not aim at selecting a "best" community structure, rather than examining the relative rating of slightly worse structures when applying the different evidence networks (based on the assumption, that randomly disturbing communities decreases their quality).

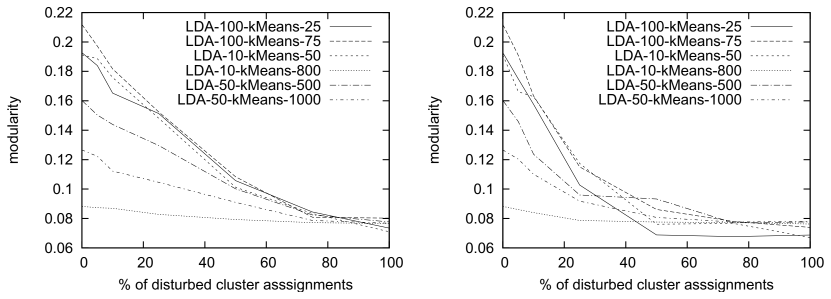
We see that the modularity on every evidence network is consistent with the level of disturbance, that is, the modularity value monotonically decreases with increasing



(a) Follower-Graph (randomized left, shuffled right).



(b) Friend-Graph (randomized left, shuffled right).

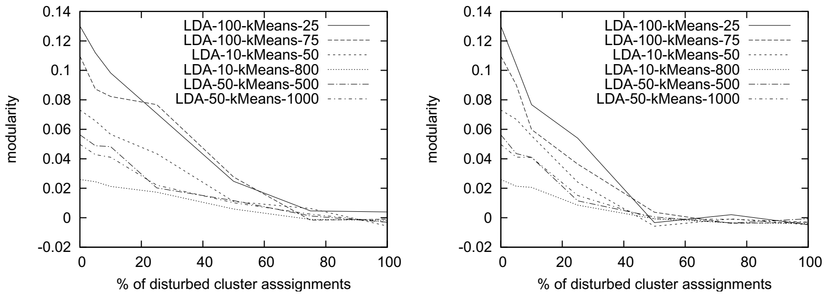


(c) Group-Graph (randomized left, shuffled right).

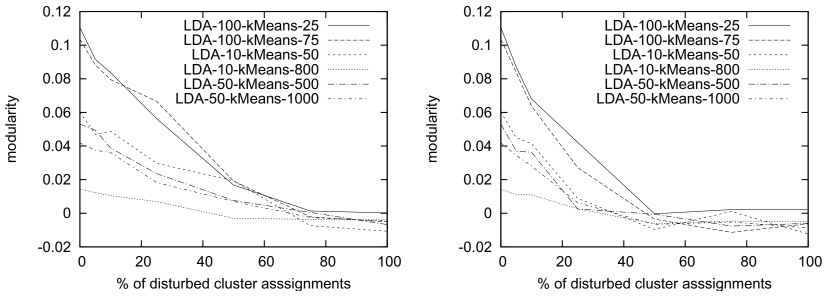
Fig. 2. Modularity calculated on different clusterings at varying levels of disturbed cluster assignments relative to explicit evidence networks

percentage of disturbed nodes. Slight deviations (e. g., looking at the alternating gradients of the Follower-Graph) are most likely statistical effects due to the limited size of the corresponding evidence network. These results are supported by the figures showing the corresponding plots for the conductance values, since lower conductance values indicate stronger clustering.

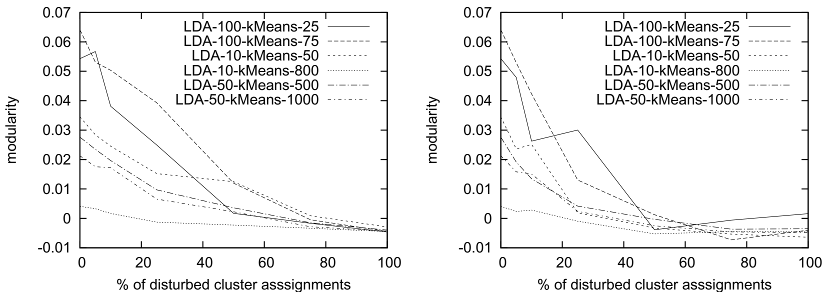
Note that conductance and modularity give precedence to different community structures. In particular, structures with many small communities are preferred according to their conductance ($k = 500, 800, 1000$), whereas smaller numbers of clusters are preferred according to their modularity (Figure 6 exemplary shows two corresponding



(a) Copy-Graph (randomized left, shuffled right).



(b) Click-Graph (randomized left, shuffled right).

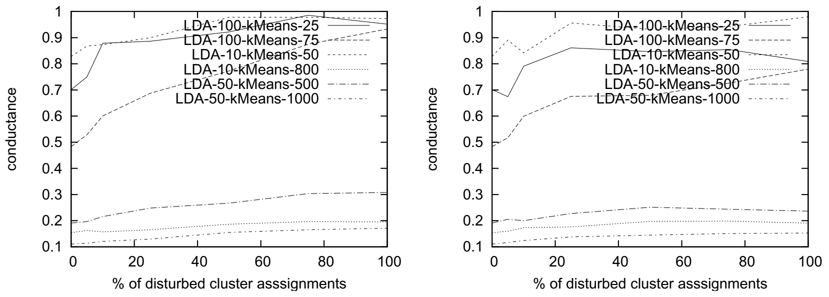


(c) Visit-Graph (randomized left, shuffled right).

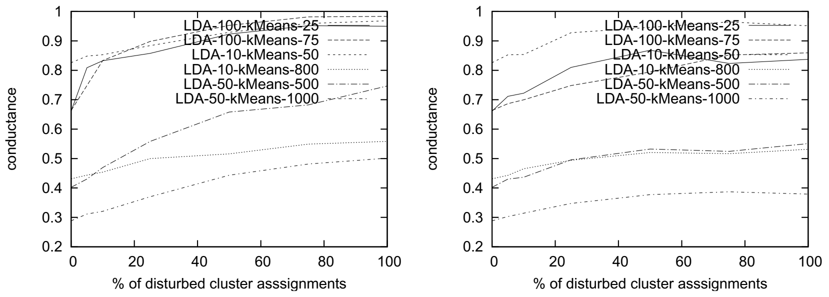
Fig. 3. Modularity calculated on different clusterings at varying levels of disturbed cluster assignments relative to implicit evidence networks

cluster size distributions). This behavior is consistent with the corresponding bias of the applied measures as discussed in [22].

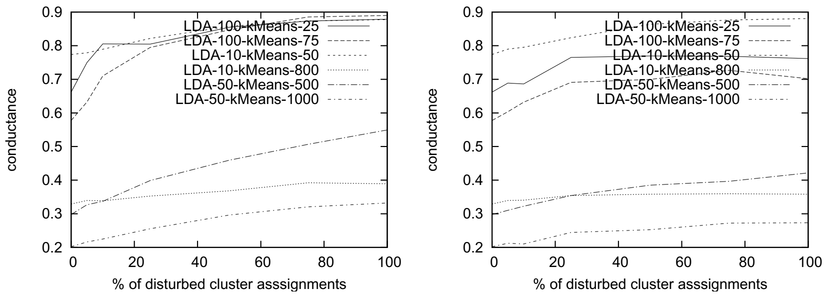
The preceding results consider the different evidence networks independently. However, we ultimately want to use the implicit networks as supplement for the sparse explicit social structures (in particular the Friend-Graph). We therefore expect the assessment of community structures applying the implicit networks to be consistent with the application of the explicit networks. This motivates the following experiment: We calculated the Pearson correlation coefficient for each of the implicit networks and one



(a) Follower-Graph (randomized left, shuffled right).



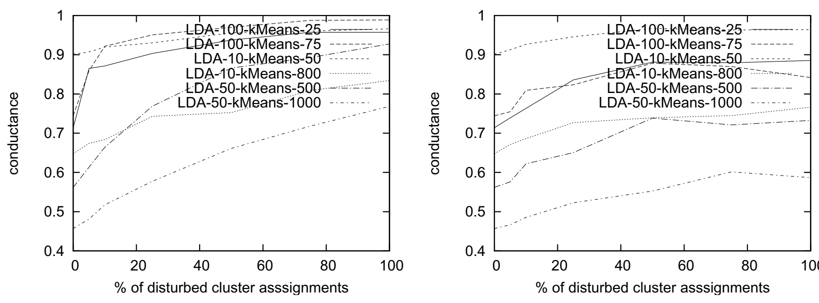
(b) Friend-Graph (randomized left, shuffled right).



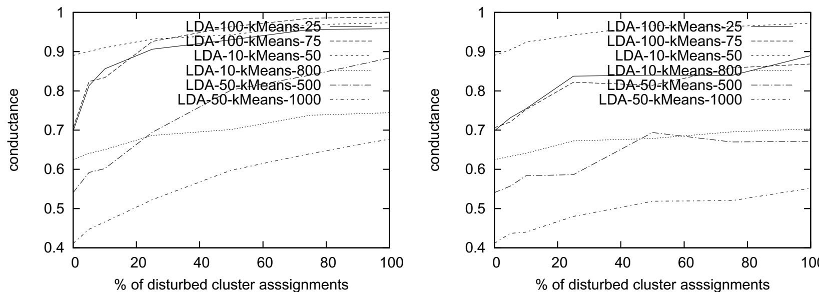
(c) Group-Graph (randomized left, shuffled right).

Fig. 4. Conductance calculated on different clusterings at varying levels of disturbed cluster assignments relative to explicit evidence networks

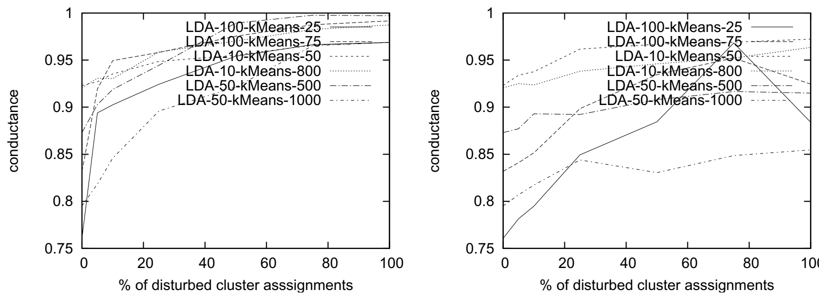
of the explicit networks. Following the paradigm of reconstructing existing social structures, the *explicit* networks yield sensible community scores (in terms of modularity and conductance). The following experimental setup aims at examining whether the corresponding community scores as induced by *implicit* networks are consistent (i. e., correlated) with the scores induced by the *explicit* networks. Table 4 shows the corresponding correlation coefficients (see Section 2.1) for modularity and conductance scores in the Friend-Graph and each of the graphs in Figures 2-5 (averaged per measure and randomization type). The averaged correlation coefficients suggest a surprisingly



(a) Copy-Graph (randomized left, shuffled right).



(b) Click-Graph (randomized left, shuffled right).



(c) Visit-Graph (randomized left, shuffled right).

Fig. 5. Conductance calculated on different clusterings at varying levels of disturbed cluster assignments relative to implicit evidence networks

high correlation between the measures calculated on the implicit networks and those calculated on the friend graph. Especially the conductance graphs show high correlation coefficients with low standard deviations. In comparison, repeating the same experiment with the group graph as the most dense existing social structure shows lower correlation coefficients with higher standard deviation, cf. Table 5.

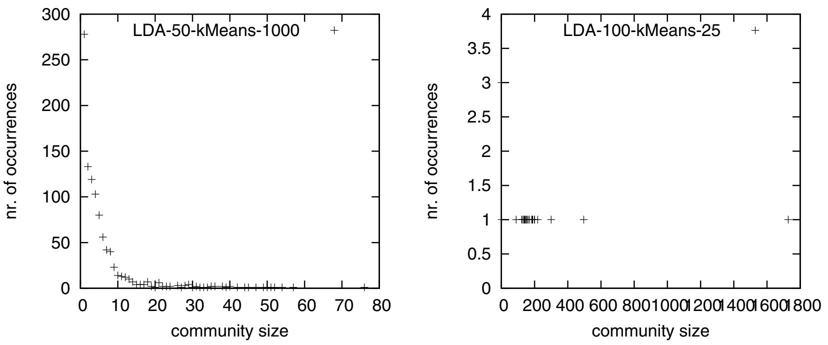


Fig. 6. Two opposed community size distributions as preferred by conductance (left) and modularity (right)

Table 4. Averaged Pearson correlation coefficient ρ_{G_i,G_2} together with it's empirical standard deviation for each of the experiments "Shuffle" (S) and "Randomize" together with the considered objective functions modularity (M) and conductance (C) on the different implicit evidence networks G_i and the friend graph G_2

Evidence Network	R/M	S/M	R/C	S/C
Follower-Graph	0.86 ± 0.17	0.90 ± 0.12	0.89 ± 0.28	0.83 ± 0.41
Group-Graph	0.91 ± 0.13	0.95 ± 0.08	1.00 ± 0.01	0.96 ± 0.17
Copy-Graph	0.82 ± 0.17	0.87 ± 0.12	0.99 ± 0.03	0.98 ± 0.09
Click-Graph	0.80 ± 0.17	0.86 ± 0.13	0.99 ± 0.04	0.98 ± 0.07
Visit-Graph	0.72 ± 0.25	0.80 ± 0.18	0.97 ± 0.06	0.98 ± 0.08

Table 5. Averaged Pearson correlation coefficient ρ_{G_i,G_3} together with it's empirical standard deviation for each of the experiments "Shuffle" (S) and "Randomize" together with the considered objective functions modularity (M) and conductance (C) on the different implicit evidence networks G_i and the group graph G_3

Evidence Network	R/M	S/M	R/C	S/C
Friend-Graph	0.91 ± 0.13	0.95 ± 0.08	1.00 ± 0.01	0.96 ± 0.17
Follower-Graph	0.72 ± 0.30	0.83 ± 0.20	0.89 ± 0.27	0.82 ± 0.40
Copy-Graph	0.67 ± 0.35	0.80 ± 0.23	0.98 ± 0.05	0.93 ± 0.29
Click-Graph	0.68 ± 0.35	0.80 ± 0.23	0.98 ± 0.04	0.94 ± 0.29
Visit-Graph	0.60 ± 0.42	0.73 ± 0.28	0.96 ± 0.07	0.93 ± 0.27

4.6 Discussion

The experimental results presented in the previous section indicate that implicit evidence networks used for assessing the quality of a community structure are surprisingly consistent with the expected behavior as formalized by the existing explicit social structures, in particular concerning the Friend-Graph. In our experiments (considering 40 models per experiment) we observed a high correlation between the quality measures calculated on the implicit and explicit networks supporting this hypothesis.

The implicit networks show a lower correlation with the group graph. At the first glance, this looks like a disappointing result. But the analysis of the group graph shows, that its properties significantly differ from typical social networks as discussed in [25,24]. Most strikingly, its degree distribution follows not a power law and its distribution of strongly connected components differs. Therefore, we obtain a ranking of the explicit graphs: It is thus more desirable to model the friend graph's behavior more closely than the group graph's.

5 Conclusions

In this paper, we have presented evidence networks for the evaluation of communities. Since explicit graph data is often sparse and does not cover the whole instance space well, evidence networks provide a viable alternative and complement to explicit networks, if available. We have discussed several possible evidence networks, and their features. The presented evaluation paradigm is based on the idea of reconstructing existing social structures for the assessment and evaluation of a given clustering. Thus, the contribution of this paper considers a new kind of data for assessing user communities in social applications is introduced and formalized as so-called evidence networks: These are thoroughly analyzed with respect to the contained community structure (cf. Section 4). It is shown that there is a strong common community structure across different networks. Our conducted experiments furthermore suggest that the different networks are not contradictory but complementary.

The context of the presented analysis is given by social applications such as social networking, social bookmarking, and social resource sharing systems, considering our own system BibSonomy[3][5] as an example. The evaluation of the presented approach using real-world data from the social resource sharing tool BibSonomy indicated the soundness of the approach considering the consistency of community structures and the applied measures. But the presented analysis is not only relevant for the evaluation of community mining techniques, but also for implementing new community detection or user recommendation algorithms, among others.

For future work, we aim to investigate, how the different evidence networks can be suitably combined into a single network. For this, we need to further analyze the individual structure of the networks, and the possible interactions. Another interesting options for further research is the improvement of the clustering algorithms on the user – tag data. An improved preprocessing of the tagging data seems a promising direction for further improving the general approach. Furthermore, we plan to extend our experiments for a larger count of networks and clusterings in order to generalize the obtained results to a broader bases.

We also plan to compare the proposed ratings of community allocations with results of different user studies, partly integrated in a live setting in the running system BibSonomy. As another direction of research, we are considering to incorporate evidence networks in the community detection process (e. g., in terms of constraints).

[3] `http://www.bibsonomy.org/`

References

1. Agresti, A.: An Introduction to Categorical Data Analysis. Wiley-Blackwell, Chichester (2007)
2. Almendral, J.A., Oliveira, J., López, L., Mendes, J., Sanjuán, M.A.: The Network of Scientific Collaborations within the European Framework Programme. Physica A: Stat. Mech. and its Applic. 384(2), 675–683 (2007)
3. Atzmueller, M., Lemmerich, F., Krause, B., Hotho, A.: Who are the Spammers? Understandable Local Patterns for Concept Description. In: Proc. 7th Conference on Computer Methods and Systems (2009)
4. Baeza-Yates, R., Tiberi, A.: Extracting Semantic Relations from Query Logs. In: Proc. 13th ACM SIGKDD Conference, p. 85. ACM, New York (2007)
5. Benz, D., Hotho, A., Jäschke, R., Krause, B., Mitzlaff, F., Schmitz, C., Stumme, G.: The Social Bookmark and Publication Management System Bibsonomy. The VLDB Journal 19(6), 849–875 (2010)
6. Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When Is Nearest Neighbor Meaningful? In: Beeri, C., Bruneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 217–235. Springer, Heidelberg (1998)
7. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet Allocation. Journal of Machine Learning Research 3, 993–1022 (2003)
8. Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., Wiener, J.: Graph Structure in the Web. Computer Networks 33(1-6), 309–320 (2000)
9. Cattuto, C., Schmitz, C., Baldassarri, A., Servedio, V., Loreto, V., Hotho, A., Grahl, M., Stumme, G.: Network Properties of Folksonomies. AI Communications 20(4), 245–262 (2007)
10. Chung, F.R.K.: Spectral Graph Theory. CBMS Reg. Conf. Series in Mathematics, vol. 92 (1997)
11. Clauset, A., Shalizi, C.R., Newman, M.E.J.: Power-Law Distributions in Empirical Data. SIAM Review 51(4) (2009)
12. Crandall, D.J., Cosley, D., Huttenlocher, D.P., Kleinberg, J.M., Suri, S.: Feedback Effects between Similarity and Social Influence in Online Communities. In: Proc. 14th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, pp. 160–168. ACM, New York (2008)
13. Diestel, R.: Graph Theory. Springer, Berlin (2006)
14. Fortunato, S., Castellano, C.: Community Structure in Graphs, arxiv:0712.2716 Chapter of Springer's Encyclopedia of Complexity and System Science (2007)
15. Gaertler, M.: Clustering. In: Brandes, U., Erlebach, T. (eds.) Network Analysis. LNCS, vol. 3418, pp. 178–215. Springer, Heidelberg (2005)
16. Karamolegkos, P.N., Patrikakis, C.Z., Doulamis, N.D., Vlacheas, P.T., Nikolakopoulos, I.G.: An Evaluation Study of Clustering Algorithms in the Scope of User Communities Assessment. Computers & Mathematics with Applications 58(8), 1498–1519 (2009)
17. Kashoob, S., Caverlee, J., Kamath, K.: Community-Based Ranking of the Social Web. In: Proceedings of the 21st ACM Conference on Hypertext and Hypermedia (2010)
18. Krause, B., Jäschke, R., Hotho, A., Stumme, G.: Logsonomy - Social Information Retrieval with Logdata. In: Proc. 19th Conf. on Hypertext and Hypermedia, pp. 157–166. ACM, New York (2008)
19. Lancichinetti, A., Fortunato, S.: Community Detection Algorithms: A Comparative Analysis, arxiv:0908.1062 (2009)
20. Leicht, E.A., Newman, M.E.J.: Community Structure in Directed Networks. Phys. Rev. Lett. 100(11) (March 2008)

21. Leskovec, J., Lang, K.J., Dasgupta, A., Mahoney, M.W.: Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters, arxiv:0810.1355 (2008)
22. Leskovec, J., Lang, K.J., Mahoney, M.W.: Empirical Comparison of Algorithms for Network Community Detection, cite arxiv:1004.3539 (2010)
23. MacQueen, J.B.: Some Methods for Classification and Analysis of MultiVariate Observations. In: Cam, L.M.L., Neyman, J. (eds.) Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297. University of California Press, Berkeley (1967)
24. Mislove, A., Marcon, M., Gummadi, K., Druschel, P., Bhattacharjee, B.: Measurement and Analysis of Online Social Networks. In: 7th ACM SIGCOMM Conf. on Internet Measurement, p. 42. ACM, New York (2007)
25. Mitzlaff, F., Benz, D., Stumme, G., Hotho, A.: Visit me, Click me, be my Friend: An Analysis of Evidence Networks of User Relationships in BibSonomy. In: HT 2010: Proc. 21st ACM Conference on Hypertext and Hypermedia, pp. 265–270. ACM, New York (2010)
26. Newman, M.E., Girvan, M.: Finding and Evaluating Community Structure in Networks. Phys. Rev. E Stat. Nonlin. Soft Matter Phys. 69(2), 1–15 (2004)
27. Newman, M.E.J.: The structure and function of complex networks. SIAM Review 45(2), 167–256 (2003)
28. Newman, M.E.J.: Detecting Community Structure in Networks. Europ. Physical J. 38 (2004)
29. Newman, M., Park, J.: Why Social Networks are different from Other Types of Networks. Physical Review E 68(3), 36122 (2003)
30. Schifanella, R., Barrat, A., Cattuto, C., Markines, B., Menczer, F.: Folks in Folksonomies: Social Link Prediction from Shared Metadata. In: Proc. 3rd ACM Int'l Conf. on Web Search and Data Mining, pp. 271–280. ACM, New York (2010)
31. Siersdorfer, S., Sizov, S.: Social Recommender Systems for Web 2.0 Folksonomies. In: HT 2009: Proc. 20th ACM Conf. on Hypertext and Hypermedia, pp. 261–270. ACM, New York (2009)
32. Szomszor, M., Cattuto, C., Van den Broeck, W., Barrat, A., Alani, H.: Semantics, Sensors, and the Social Web: The Live Social Semantics Experiments. In: The Semantic Web: Research and Applications, pp. 196–210 (2010)

Towards Adjusting Mobile Devices to User's Behaviour

Peter Fricke[1], Felix Jungermann[1], Katharina Morik[1], Nico Piatkowski[1], Olaf Spinczyk[2], Marco Stolpe[1], and Jochen Streicher[2]

[1] TU Dortmund University,
Artificial Intelligence Group
Baroper Strasse 301, Dortmund, Germany
{fricke,jungermann,morik,piatkowski,stolpe}@ls8.cs.tu-dortmund.de
http://www-ai.cs.tu-dortmund.de
[2] TU Dortmund University,
Embedded System Software Group
Otto-Hahn-Strasse 16, Dortmund, Germany
{olaf.spinczyk,jochen.streicher}@tu-dortmund.de
http://ess.cs.uni-dortmund.de

Abstract. Mobile devices are a special class of resource-constrained embedded devices. Computing power, memory, the available energy, and network bandwidth are often severely limited. These constrained resources require extensive optimization of a mobile system compared to larger systems. Any needless operation has to be avoided. Time-consuming operations have to be started early on. For instance, loading files ideally starts before the user wants to access the file. So-called prefetching strategies optimize system's operation. Our goal is to adjust such strategies on the basis of logged system data. Optimization is then achieved by predicting an application's behavior based on facts learned from earlier runs on the same system. In this paper, we analyze system-calls on operating system level and compare two paradigms, namely server-based and device-based learning. The results could be used to optimize the runtime behaviour of mobile devices.

Keywords: Mining system calls, ubiquitous knowledge discovery.

1 Introduction

Users demand mobile devices to have long battery life, short application startup time, and low latencies. Mobile devices are constrained in computing power, memory, energy, and network connectivity. This conflict between user expectations and resource constraints can be reduced, if we tailor a mobile device such that it uses its capacities carefully for exactly the user's needs, i.e., the services, that the user wants to use. Predicting the user's behavior given previous behavior is a machine learning task. For example, based on the learning of most often used file path components, a system may avoid unnecessary probing of files and could intelligently prefetch files. Prefetching those files, which soon will be accessed by the system, leads to a grouping of multiple scattered I/O requests to a batched one and, accordingly, conservation of energy. The resource restrictions of mobile devices motivate the application of machine learning for predicting user behavior. At the same time, machine learning dissipates resources. There are four critical resource constraints:

– Data gathering: logging user actions uses processing capacity.
– Data storage: the training and test data as well as the learned model use memory.
– Communication: if training and testing is performed on a central server, sending data and the resulting model uses the communication network.
– Response time: the prediction of usage, i.e., the model application, has to happen in short real-time.

The dilemma of saving resources at the device through learning which, in turn, uses up resources, can be solved in several ways. Here, we set aside the problem of data gathering and its prerequisites on behalf of operation systems for embedded systems [15] [24] [4]. This is an important issue in its own right. Regarding the other restrictions, especially the restriction of memory, leads us to two alternatives.

Server-based learning: The learning of usage profiles from data is performed on a server and only the resulting model is communicated back to the device. Learning is less restricted in runtime and memory consumption. Just the learned model must obey the runtime and communication restrictions. Hence, a complex learning method is applicable. Figure 1 shows this alternative.

Device-based learning: The learning of usage profiles on the device is severely restricted in complexity. It does not need any communication but requires training data to be stored. Data streaming algorithms come into play in two alternative ways. First, descriptive algorithms incrementally build-up a compact way to store data. They do not classify or predict anything. Hence, in addition, simple methods are needed that learn from the aggregated compact data. Second, simple online algorithms predict usage behavior in realtime. The latter option might only be possible if specialized hardware is used, e.g., General Purpose GPUs. Figure 2 shows this alternative.

In this paper, we want to investigate the two alternatives using logged system calls. Server-based learning is exemplified by predicting file-access patterns in order to enhance prefetching. It is an open question whether structural models are demanded for the prediction of user behavior on the basis of system calls, or simpler models such as Naive Bayes suffice. Should the sequential nature of system calls be taken into account by the algorithm? Or is it sufficient to encode the sequences into the features? Or should features as well as algorithm be capable of explicitly utilizing sequences? In order to address these questions, we

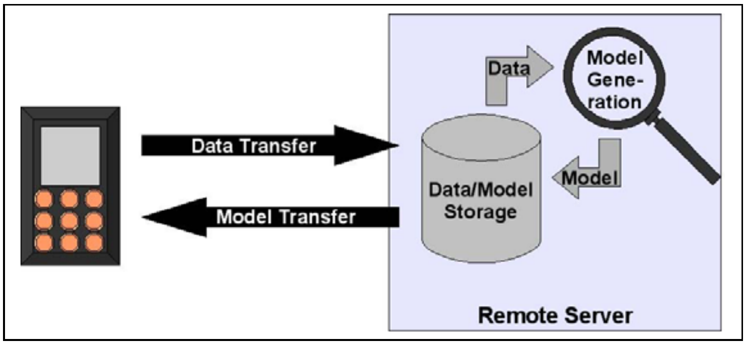


Fig. 1. Server-based Architecture

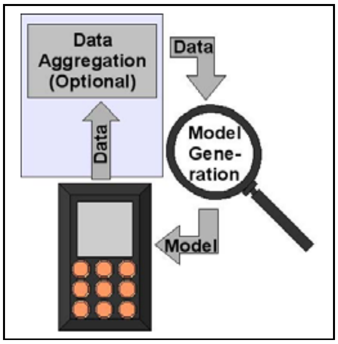


Fig. 2. Device-based Architecture

investigate the use of two extremes: Conditional Random Fields (CRF) – which use sequential information – and Naive Bayes (NB) – which ignores sequential dependencies among the labels. In particular, we inspect their memory consumption and runtime, both, for training and applying the learned function. Section 2 presents the study of server-based learning for ubiquitous devices. We derive the learning task from the need of enhancing prefetching strategies, describe the log data used, and present the learning results together with resource consumptions of NB and CRF.

Device-based learning is exemplified by recognizing applications from system calls in order to prevent fraud. We apply the data streaming algorithm Hierarchical Heavy Hitters (HHH) yielding a compact data structure for storage. Using these, the simple kNN method classifies systems calls. In particular, we investigate how much HHH compress data. Section 3 presents the study of device-based learning using a streaming algorithm for storing compact data. We conclude in Section 4 by indicating related and future work.

2 Server-Based Learning

We present the first case-study, where log data is stored and analyzed on a server. The acquisition is described in Section 2.2 and the data itself in Section 2.3. Learning aims at predicting file access in order to prefetch files (see Section 2.1). The learning methods NB and CRF are introduced shortly in Section 2.4 and Section 2.5, respectively. The results are shown in Section 2.6.

2.1 File-Access Pattern Prediction

A prediction of file-access patterns is of major importance for the performance and resource consumption of system software. For example, the Linux operating system uses a large "buffer cache" memory for disk blocks. If a requested disk block is already stored in the cache (*cache hit*), the operating system can deliver it to the application much faster and with less energy consumption than otherwise (*cache miss*). In order to manage the cache the operating system has to implement two strategies, block replacement and prefetching. The *block replacement* strategy is consulted upon a cache miss: a new block has to be inserted into the cache. If the cache is already full, the strategy has to decide which block has to be replaced. The most effective victim is the one with the longest forward distance, i.e. the block with the maximum difference between now and the time of the next access. This requires to know or guess the future sequence of cache access. The *prefetching* strategy proactively loads blocks from disk into the cache, even if they have not been requested by an application, yet. This often pays off, because reading a bigger amount of blocks at once is more efficient than multiple read operations. However, prefetching should only be performed if a block will be needed in the near future. For both strategies, block replacement and prefetching, a good prediction of future application behavior is crucial.

Linux and other operating systems still use simple heuristic implementations of the buffer cache management strategies. For instance, the prefetching code in Linux [2] continuously monitors read operations. As long as a file is accessed sequentially the read ahead is increased. Certain upper and lower bounds restrict the risk of mispredictions. This heuristics has two flaws:

– No prefetching is performed *before* the first read operation on a specific file, e.g., after "open", or even earlier.
– The strategy is based on assumptions on typical disk performance and buffer cache sizes, in general. However, these assumptions might turn out to be wrong in certain application areas or for certain users.

Prefetching based on machine learning avoids both problems. Prefetching can already be performed when a file is opened. It only depends on the prediction that the file will be read. The prediction is based on empirical data and not on mere assumptions. If the usage data change, the model changes, as well.

2.2 System Call Data Acquisition

Logging system calls in the Linux kernel does not only require instrumentation, but also a mechanism to transport the collected data out of the kernel space. Since the kernel's own memory pages cannot be swapped out of the main memory, kernel memory is usually kept as small as possible. Therefore, data transport has to happen frequently and should thus be efficient.

A convenient tool for this purpose is *SystemTap* [9], which allows the use of an event-action language for kernel instrumentation. The most important language element is the *probe*, which consists of two parts: The first part describes events of interest, like system calls, in a declarative manner. The second part is a corresponding handler, which is written in a C-like typesafe language. When at least one of the specified events of a probe occurs, the handler is executed. The handlers usually write collected data into an in-kernel buffer, but they may also preprocess or accumulate data first. Depending on the type of an event, the handler has access to context information, like function parameters and return values. Global kernel data, like the current thread ID, is always accessible.

SystemTap provides a compiler, which translates the probe definitions into a loadable kernel extension module. As soon as the module is loaded, it instruments all associated points in the kernel machine code with calls to corresponding probe handlers. After that, Systemtap's runtime system constantly moves the generated data from kernel to user space.

Data Acquisition on Android-Based Devices. Our mobile system call data source was an *HTC Desire* smartphone, which is based on the mobile operating system Android.

Although Android has a Java-based application layer, the layers below contain all essential parts of an embedded Linux system. Besides the kernel itself, there are also all standard libraries and tools that are required to run SystemTap. However, the kernel included in a device vendor's standard installation usually has several features disabled that are essential for using SystemTap, e.g., support for instrumentation.

To build a SystemTap-enabled kernel, we used *Cyanogenmod*[1], an Android-based software distribution, which exists in various device-specifically customized variants. The original Android sources contain only a generic Linux kernel, which does not necessarily support all the hardware components of a specific device.

SystemTap's workflow for embedded devices is designed to perform as much work as possible on an external, more powerful machine (the *host*), leaving only necessary parts on the mobile device (the *target*). The probe definitions are compiled on the *host*, which contains most of the SystemTap software, as well as the debug information for the target's kernel. The target contains merely the runtime and, of course, the compiled instrumentation modules.

The average amount of log data per day was only 11 MiB in size. Thus, we could store the entire log file on the device's internal flash drive. In order to

[1] We used Cyanogenmod 7.0.0, which is based on Android 2.3.3. The kernel release was 2.6.32.28. Cyanogenmod can be downloaded from `http://www.cyanogenmod.com/`

preserve the user's privacy, it is planned to encrypt the log data in future experiments. The instrumentation did not have any user-observable impact on performance and responsiveness during typical operation (e.g., phoning, writing text messages, playing audio files, and browsing the web).

2.3 System Call Data for Access Prediction

We logged streams of system calls of type FILE on a desktop system as well as an Android-based mobile phone. System calls consist of various typical subsequences, each starting with an `open`- and terminating with a `close`-call, like those shown in Figure 3 and 4. We collapsed such sub-sequences to one observation and assign the class label

- **full**, if the opened file was read from the first seek (if any) to the end,
- **read**, if the opened file was randomly accessed and
- **zero**, if the opened file was not read after all.

```
1,open,1812,179,178,201,200,firefox,/etc/hosts,524288,438,7 : 361, full
2,read,1812,179,178,201,200,firefox,/etc/hosts,4096,361
3,read,1812,179,178,201,200,firefox,/etc/hosts,4096,0
4,close,1812,179,178,201,200,firefox,/etc/hosts
```

Fig. 3. A sequence of system calls to *read* a file. The data layout is: timestamp, syscall, thread-id, process-id, parent, user, group, exec, file, parameters (optional) : read bytes, label (optional)

```
1,open,14,14,1,25,100,gconfd-2,/path/to/gconf.xml.new,65,384,47 : 0, zero
2,llseek,14,14,1,25,100,gconfd-2,/path/to/gconf.xml.new,1,0,96
3,write,14,14,1,25,100,gconfd-2,/path/to/gconf.xml.new,697
4,close,14,14,1,25,100,gconfd-2,47
```

Fig. 4. A sequence of system calls to *write* some blocks to a file. The data layout is the same as for Figure 3.

We propose the following generalization of obtained filenames. If a file is regular, we remove anything except the filename extension. Directory names are replaced by "DIR", except for paths starting with "/tmp" – those are replaced by "TEMP". Any other filenames are replaced by "OTHER". This generalization of filenames yields good results in our experiments. Volatile information like thread-id, process-id, parent-id and system-call parameters is dropped, and consecutive

Table 1. Snippet of the preprocessed dataset (the marked row corresponds to the open call of Fig. 3)

user	group	exec	file	label
201	200	firefox-bin	cookies.sqlite-journal	zero
201	200	firefox-bin	default	zero
201	200	firefox-bin	hosts	full
201	200	firefox-bin	hosts	full
201	200	multiload-apple	mtab	full
102	200	kmail	png	zero

Table 2. Cost matrix

predicted\true	full	zero	read
full	0	2	1
zero	5	0	4
read	4	2	0

Table 3. Snippet of the final dataset using two features

exec	file	label
?_firefox-bin	?_?_cookies.sqlite-journal	zero
firefox-bin_firefox-bin	?_cookies.sqlite-journal_default	zero
firefox-bin_firefox-bin	cookies.sqlite-journal_default_hosts	full
firefox-bin_firefox-bin	default_hosts_hosts	full
?_multiload-apple	?_?_mtab	full
?_kmail	?_?_png	zero

observations are compound to one sequence if they belong to the same process. The resulting dataset consists of 673887 observations for the desktop log data and 18257 for the Android log data. These are aggregated into 80661 and 3328 sequences, respectively. A snippet[2] is shown in Table 1.

We used two feature sets for the given task. The first encodes information about sequencing as features, resulting in 24 features, namely f_t, f_{t-1}, f_{t-2}, f_{t-2}/f_{t-1}, f_{t-1}/f_t, $f_{t-2}/f_{t-1}/f_t$, with $f \in \{user, group, exec, file\}$. The second feature set simply uses two features $exec_{t-1}/exec_t$ and $file_{t-2}/file_{t-1}/file_t$ as its only features – an excerpt of the dataset using these two features is shown in Table 3.

Errors in predicting the types of access result in different degrees of failure. Predicting a partial caching of a file, if just the rights of a file have to be changed, is not as problematic as predicting a partial read if the file is to be read completely. Hence, we define a cost-matrix (see Table 2) for the evaluation of our approach. For further research the values used in this matrix might have to be readjusted based on results of concrete experiments on mobile devices or simulators.

[2] The final dataset is available at:
`http://www-ai.cs.tu-dortmund.de/PUBDOWNLOAD/MUSE2010`

2.4 Naive Bayes Classifier

The Naive Bayes classifier (cf. [12]) assigns labels $y \in Y$ to examples $x \in X$. Each example is a vector of m attributes written here as x_i, where $i = 1...m$. The probability of a label given an example is according to the Bayes Theorem:

$$p(Y|x_1, x_2, ..., x_m) = \frac{p(Y)p(x_1, x_2, ..., x_m|Y)}{p(x_1, x_2, ..., x_m)} \tag{1}$$

Domingos and Pazzani [8] rewrite eq. (1) and define the *Simple Bayes Classifier* (SBC):

$$p(Y|x_1, x_2, ..., x_m) = \frac{p(Y)}{p(x_1, x_2, ..., x_m)} \prod_{j=1}^{n} p(x_j|Y) \tag{2}$$

The classifier delivers the most probable class Y for a given example $x = x_1 \dots x_m$:

$$\arg\max_{Y} p(Y|x_1, x_2, ..., x_m) = \frac{p(Y)}{p(x_1, x_2, ..., x_m)} \prod_{j=1}^{m} p(x_j|Y) \tag{3}$$

The term $p(x_1, x_2, ..., x_m)$ can be neglected in eq. (3) because it is a constant for every class $y \in Y$. The decision for the most probable class y for a given example x just depends on $p(Y)$ and $p(x_i|Y)$ for $i = 1 \dots m$. These probabilities can be calculated after one run on the training data. So, the training runtime is $\mathcal{O}(n)$, where n is the number of examples in the training set. The number of probabilities to be stored during training are $|\mathcal{Y}| + (\sum_{i=1}^{m} |\mathcal{X}_j| * |\mathcal{Y}|)$, where $|\mathcal{Y}|$ is the number of classes and $|\mathcal{X}_i|$ is the number of different values of the ith attribute. The storage requirements for the trained model are $\mathcal{O}(mn)$.

It has often been shown that SBC or NBC perform quite well for many data mining tasks [8,13,10].

2.5 Linear-Chain Conditional Random Fields

Linear-chain Conditional Random Fields, introduced by Lafferty et al. [14], can be understood as discriminative, sequential version of Naive Bayes Classifiers. The conditional probability for an actual sequence of labels $\mathbf{y_1}, \mathbf{y_2}, ..., \mathbf{y_m}$, given a sequence of observations $\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_m}$ is modeled as an exponential family. The underlying assumption is that a class label at the current timestep t just depends on the label of its direct ancestor, given the observation sequence. Dependency among the observations is not explicitly represented, which allows the use of rich, overlapping features. Equation 4 shows the model formulation of linear-chain CRF

$$p_\lambda(Y = \mathbf{y}|X = \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \exp\left(\sum_k \lambda_k f_k(y_t, y_{t-1}, \mathbf{x})\right) \tag{4}$$

with the observation-sequence dependent normalization factor

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \prod_{t=1}^{T} \exp\left(\sum_{k} \lambda_k f_k\left(y_t, y_{t-1}, \mathbf{x}\right)\right) \tag{5}$$

The sufficient statistics or *feature functions* f_k are most often binary indicator functions which evaluate to 1 only for a single combination of class label(s) and attribute value. The parameters λ_k can be regarded as weights or scores for this feature functions. In linear-chain CRF, each attribute value usually gets $|\mathcal{Y}|+|\mathcal{Y}|^2$ parameters, that is one score per state-attribute pair as well as one score for every transition-attribute triple, which results in a total of $\sum_{i=1}^{m} |\mathcal{X}_i| \left(|\mathcal{Y}| + |\mathcal{Y}|^2\right)$ model parameters, where $|\mathcal{Y}|$ is the number of classes, m is the number of attributes and $|\mathcal{X}_i|$ is the number of different values of the ith attribute. Notice that the feature functions explicitly depend on the whole observation-sequence rather than on the attributes at time t. Hence, it is possible and common to involve attributes of preceding as well as following observations from the current sequence into the computation of the total score $\exp\left(\sum_k \lambda_k f_k\left(y_t, y_{t-1}, \mathbf{x}\right)\right)$ for the transition from y_{t-1} to y_t given $\mathbf{x}$.

The parameters are usually estimated by the maximum-likelihood method, i.e., maximizing the conditional likelihood (Eq. 6) by quasi-Newton [16], [21], [17] or stochastic gradient methods [27], [19], [20].

$$\mathcal{L}(\lambda) = \prod_{i=1}^{N} p_\lambda(Y = \mathbf{y}^{(i)} | X = \mathbf{x}^{(i)}) \tag{6}$$

The actual class prediction for an unlabeled observation-sequence is done by the Viterbi algorithm known from Hidden Markov Models [23], [18].

Although CRF in general allow to model arbitrary dependencies between the class labels, efficient exact inference can solely be done for linear-chain CRF. This is no problem here, because they match the sequential structure of our system-call data, presented in section 2.3.

2.6 Results of Server-Based Prediction

Tables 4 to 9 are showing the results on the desktop dataset and Tables 10 to 15 for Android, respectively. Comparing the prediction quality of the simple NB models and the more complex CRF models, surprisingly, the CRF are only slightly better when using the two best features. CRF outperforms NB when using all features. These two findings indicate that the sequence information is not as important as we expected. Neither encoding the sequence into features nor applying an algorithm which is made for sequential information outperforms a simple model. The Tables show that precision, recall, accuracy, and misclassification cost are quite homogeneous for CRF, but vary for NB. In particular, the precision of predicting "read" and the recall of class "zero" differs from the numbers for the other classes, respectively. This makes CRF more reliable.

The results on the Android log data resemble those on the desktop log except for the precision on label "full". This might be caused by the lower number of recorded system calls as well as different label proportions.

Inspecting resource consumption, we stored models of the two methods for both feature sets and for various numbers of examples to show the practical storage needs of the methods. Table 16 presents the model sizes of the naive Bayes classifier on both feature sets and for various example set sizes. We used the popular open source data mining tool *RapidMiner*[3] for these experiments. Table 16 also shows the model sizes of CRF on both feature sets and various example set sizes.

We used the open source CRF implementation *CRF++*[4] with L_2-regularization, $\sigma = 1$ and L-BFGS optimizer in all CRF experiments. Obviously, the storage needs for a model produced by a NB classifier are lower than those for a CRF model. This is the price to be paid for more reliable prediction quality. CRF don't scale-up well. Considering training time, the picture becomes worse. Table 17 shows the training time of linear-chain or HMM-like CRF consuming orders of magnitude more time than NB.

3 Device-Based Learning

In this section, we present the second case-study, where streams of log data are processed in order to store patterns of system use. The goal is to aggregate the streaming system data. A simple learning method might then use the aggregated data. The method of Hierarchical Heavy Hitters (HHH) is defined in Section 3.1. The log data are shown in Section 3.2. For the comparison of different sets of HHH, we present a distance measure that allows for clustering or classifying sets of HHH. In addition to the quality of our HHH application, its resource consumption is presented in Section 3.3.

3.1 Hierarchical Heavy Hitters

The *heavy hitter problem* consists of finding all frequent elements and their frequency values in a data set. According to Cormode [5], given a (multi)set S of size N and a threshold $0 < \phi < 1$, an element e is a *heavy hitter* if its frequency $f(e)$ in S is not smaller than $\lfloor \phi N \rfloor$. The set of heavy hitters is then $HH = \{e | f(e) \geq \lfloor \phi N \rfloor\}$.

If the elements in S originate from a hierarchical domain D, one can state the following problem [5]:

Definition 1 (HHH Problem). *Given a (multi)set S of size N with elements e from a hierarchical domain D of height h, a threshold $\phi \in (0, 1)$ and an error parameter $\epsilon \in (0, \phi)$, the* Hierarchical Heavy Hitter Problem *is that of identifying prefixes $P \in D$, and estimates f_p of their associated frequencies, on the first N consecutive elements S_N of S to satisfy the following conditions:*

[3] RapidMiner is available at: `http://www.rapidminer.com`

[4] CRF++ is available at: `http://crfpp.sourceforge.net/`

Table 4. Result of Naive Bayes Classifier on the best two features, 10x10-fold cross-validated, accuracy: 94.45 ± 0.0, missclassification costs: 0.152 ± 0.01

predicted\true	**full**	**zero**	**read**	**prec.**
full	1427467	19409	3427	98.43
zero	12541	2469821	40258	97.91
read	80872	217380	2467695	89.22
recall	93.86	91.25	98.26	

Table 5. Result of Naive Bayes Classifier on all 24 features, 10x10-fold cross-validated, accuracy: 91.84 ± 0.0, missclassification costs: 0.218 ± 0.02

full	**zero**	**read**	**prec.**
1426858	21562	22717	96.99
15392	2371009	97566	95.45
78630	314039	2391097	85.89
93.82	87.60	95.21	

Table 6. Result of HMM-like CRF on the best two features, 10x10-fold cross-validated, accuracy: 95.49 ± 0.0, missclassification costs: 0.150 ± 0.0

predicted\true	**full**	**zero**	**read**	**prec.**
full	1446242	7123	29051	97.56
zero	19452	2639097	133007	94.54
read	55186	60390	2349322	95.31
recall	95.09	97.51	93.55	

Table 7. Result of HMM-like CRF on all 24 features, 10x10-fold cross-validated, accuracy: 95.70 ± 0.0, missclassification costs: 0.143 ± 0.0

full	**zero**	**read**	**prec.**
1450147	8335	25629	97.71
14563	2639724	126403	94.93
56170	58551	2359348	95.36
95.35	97.53	93.95	

Table 8. Result of linear-chain CRF on the best two features, 10x10-fold cross-validated, accuracy: 96.79 ± 0.0, missclassification costs: 0.112 ± 0.0

predicted\true	**full**	**zero**	**read**	**prec.**
full	1467440	4733	7503	99.17
zero	10883	2659294	108340	95.71
read	42557	42583	2395537	96.57
recall	96.49	98.25	95.39	

Table 9. Result of linear-chain CRF on all 24 features, 10x10-fold cross-validated, accuracy: 96.89 ± 0.0, missclassification costs: 0.110 ± 0.0

full	**zero**	**read**	**prec.**
1468095	4117	5022	99.38
10306	2662966	107859	95.75
42479	39527	2398499	96.69
96.53	98.39	95.51	

– *accuracy:* $f_p^* - \varepsilon N \leq f_p \leq f_p^*$*, where* f_p^* *is the true frequency of* p *in* S_N*.*
– *coverage: all prefixes* $q \notin P$ *satisfy* $\phi N > \sum f(e) : (e \preceq q) \wedge (\nexists p \in P : e \preceq p)$*.*

Here, $e \preceq p$ means that element e is *generalizable* to p (or $e = p$). For the extended multi-dimensional heavy hitter problem introduced in [6], elements can be multi-dimensional d-tuples of hierarchical values that originate from d different hierarchical domains with depth $h_i, i = 1, \ldots, d$. There exist two variants of algorithms for the calculation of multi-dimensional HHHs: Full Ancestry and Partial Ancestry, which we have both implemented. For a detailed description of these algorithms, see [7].

Table 10. Result of Naive Bayes Classifier on best two features, 10x10-fold cross-validated, accuracy: 96.27 ± 0.03, missclassification costs: 0.08 ± 0.0

predicted\true	full	zero	read	prec.
full	20322	3027	123	86.57
zero	290	108919	463	99.31
read	108	2794	46524	94.12
recall	98.07	94.92	98.75	

Table 11. Result of Naive Bayes Classifier on all 24 features, 10x10-fold cross-validated, accuracy: 96.29 ± 0.05, missclassification costs: 0.09 ± 0.0

full	zero	read	prec.
20041	2937	106	86.81
546	109364	610	98.95
133	2439	46394	94.74
96.72	95.31	98.48	

Table 12. Result of HMM-like CRF on the best two features, 10x10-fold cross-validated, accuracy: 97.07 ± 0.0, missclassification costs: 0.077 ± 0.0

predicted\true	full	zero	read	prec.
full	19846	3078	305	85.43
zero	722	111274	658	98.77
read	162	408	46147	98.77
recall	95.73	96.96	97.95	

Table 13. Result of HMM-like CRF on all 24 features, 10x10-fold cross-validated, accuracy: 97.97 ± 0.0, missclassification costs: 0.050 ± 0.0

full	zero	read	prec.
20279	2745	53	87.87
344	111890	320	99.41
107	125	46737	99.50
97.82	97.49	99.20	

Table 14. Result of linear-chain CRF on the best two features, 10x10-fold cross-validated, accuracy: 97.62 ± 0.0, missclassification costs: 0.058 ± 0.0

predicted\true	full	zero	read	prec.
full	20145	2994	246	86.14
zero	481	111572	313	99.29
read	104	194	46551	99.36
recall	97.17	97.22	98.81	

Table 15. Result of linear-chain CRF on all 24 features, 10x10-fold cross-validated, accuracy: 98.00 ± 0.0, missclassification costs: 0.047 ± 0.0

full	zero	read	prec.
20337	2710	71	87.97
173	111813	233	99.63
220	237	46806	99.03
98.10	97.43	99.35	

3.2 System Call Data for HHH

The kernel of current Linux operating systems offers about 320 different types of system calls to developers. Having gathered all system calls made by several applications, we observed that about 99% of all calls belonged to one of the 54 different call types shown in Tab. 18. The functional categorization of system calls into five groups is due to [22]. We focus on those calls only, since the remaining 266 call types are contained in only 1% of the data and therefore can't be frequent.

Table 16. Storage needs (in kB) of the naive Bayes (nB), the HMM-like CRF (CRF++ (HMM)) and the linear-chain CRF (CRF++) classifier model produced by *RapidMiner* on different numbers of sequences and attributes

#Att.\#Seq.	**0**	**67k**	**135k**	**202k**	**270k**	**337k**	**404k**	**472k**	**539k**	**606k**	**674k**
2 nB	244	248	251	253	256	255	256	257	257	256	256
24 nB	548	561	571	577	582	585	588	590	590	585	585
2 CRF++ (HMM)	5	247	366	458	490	512	569	592	614	634	649
24 CRF++ (HMM)	12	615	878	1102	1170	1216	1367	1420	1463	1521	1551
2 CRF++	6	523	776	978	1043	1089	1213	1260	1299	1345	1378
24 CRF++	19	1339	1914	2415	2559	2652	2988	3095	3184	3303	3365

Table 17. Training time (in seconds) of the naive Bayes (nB), the HMM-like CRF (CRF++ (HMM)) and the linear-chain CRF (CRF++) classifier model produced by *RapidMiner* on different numbers of sequences and attributes

#Att.\#Seq.	**0**	**67k**	**135k**	**202k**	**270k**	**337k**	**404k**	**472k**	**539k**	**606k**	**674k**
2 nB	< 1	< 1	< 1	< 1	1	< 1	< 1	< 1	< 1	< 1	< 1
24 nB	< 1	< 1	< 1	1	< 1	1	1	1	1	2	1
2 CRF++ (HMM)	< 1	9.09	28.56	44.08	60.1	75.76	107.28	127.04	149.95	165.94	199.2
24 CRF++ (HMM)	< 1	27.92	55.9	103.24	153.53	160.33	230.7	273.29	232.84	309.19	317.62
2 CRF++	< 1	16.69	50.23	85.18	113.21	145.96	173.56	200.98	234.65	260.56	325.54
24 CRF++	< 1	41.06	105.29	156.67	296.31	300.83	343.28	433.03	440.88	463.84	632.96

HHHs can handle values that have a hierarchical structure. We have utilized this expressive power by representing system calls as tuples of up to three hierarchical feature values originating from corresponding taxonomies: *system call types*, *file paths* and *call sequences*.

The groups introduced in Tab. 18 form the top level of the taxonomy for the system call types (see Fig. 5). The `socket` call is a child of group COMM and FILE is the parent of calls like `open` and `fcntl64`. Subtypes of system calls can be defined by considering the possible values of their parameters. For example, the `fcntl64` call which operates on file descriptors has *fd*, *cmd* and *arg* as its parameters. We have divided the 16 different nominal values of the *cmd* parameter into seven groups — `notify`, `dflags`, `duplicate`, `sig`, `lock`, `fflags` and `lease` — that have become the children of the `fcntl64` system call in our taxonomy (see Fig. 5). One may further divide `fcntl64` calls of subtype `fflags` by the values `F_SETFL` and `F_GETFL` of the *arg* parameter. In the same way, we defined parents and children for each of the 54 call types and their parameters.

Albeit the taxonomy we present here already yields promising results in our experiments, we consider it to be an open research question how to find a categorization of system calls that fits a given learning task.

The hierarchical variable *file path* is defined whenever a system call accesses a file system path. Its hierarchy comes naturally along with the given file path hierarchy of the file system. The *call sequence* variable expresses the temporal order of calls within a process. The directly preceding call is the highest, less recent calls are at deeper levels of the hierarchy. The information which is kept in a sequence are the names of the system calls.

Table 18. We focus on 54 system call types which are functionally categorized into five groups. FILE: file system operations, COMM: communication, PROC: process and memory management, INFO: informative calls, DEV: operations on devices.

FILE	COMM	PROC	INFO	DEV
`open`	`recvmsg`	`mmap2`	`access`	`ioctl`
`read`	`recv`	`munmap`	`getdents`	
`write`	`send`	`brk`	`getdents64`	
`lseek`	`sendmsg`	`clone`	`clock_gettime`	
`_llseek`	`sendfile`	`fork`	`gettimeofday`	
`writev`	`sendto`	`vfork`	`time`	
`fcntl`	`rt_sigaction`	`mprotect`	`uname`	
`fcntl64`	`pipe`	`unshare`	`poll`	
`dup`	`pipe2`	`execve`	`fstat`	
`dup2`	`socket`	`futex`	`fstat64`	
`dup3`	`accept`	`nanosleep`	`lstat`	
`close`	`accept4`		`lstat64`	
			`stat`	
			`stat64`	
			`inotify_init`	
			`inotify_init1`	
			`readlink`	
			`select`	

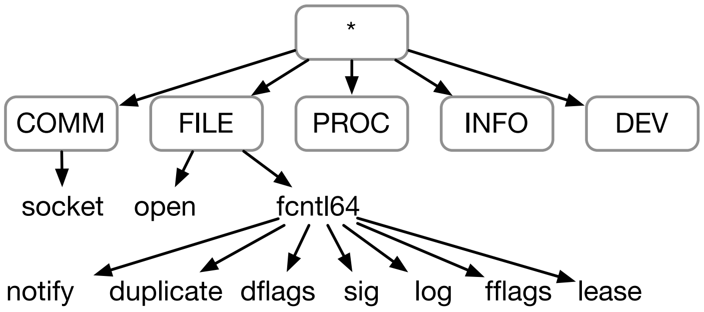


Fig. 5. Parts of the taxonomy we defined for the hierarchical variable *system call type*

Collected Data. We have implemented a parser that reads log files and translates them into hierarchical value tupels according to the three taxonomies.

We collected system call data from eleven applications (like Firefox, Epiphany, NEdit, XEmacs) shown in Tab. 19 under Ubuntu Linux (kernel 2.6.26, 32 bit). For each application, we logged five times five minutes and five times ten minutes of system calls if they belonged to one of the 54 types shown in Tab. 18, resulting in a whole of 110 log files comprising about 23 million of lines (1.8 GiB).

Table 19. List of applications for which system calls were logged

Application	Version	Function
Firefox	3.0.15	Webbrowser
top	3.2.7	Display of running processes
Rhythmbox	0.12.0	Audio player
Geyes	2.26.1	Eyes following mouse pointer
NEdit	5.5	Text editor
Vinagre	2.26.1	Remote control
XEmacs	21.4.21	Text editor
Kate	3.2.2	Text editor
xterm	241	Terminal emulator
Tomboy	0.14.0	Editor for notes
Epiphany	2.26.1	Webbrowser

Table 20. Memory consumption (number of stored tupels), run-time (milliseconds) and similarity to exact solution of the Full Ancestry (FA) and Partial Ancestry (PA) algorithms ($\varepsilon = 0.0005$, $\phi = 0.002$). Minimum (Min), maximum (Max) and average (Avg) values were calculated over measurements for the first log file of all eleven applications with varying dimensionality of the element tupels (T = *system call type* hierarchy, P = *file path* hierarchy, S = *call sequence* hierarchy).

		Memory			**Run-time**			**Similarity**	
		Min	Max	Avg	Min	Max	Avg	Avg	Dev
	T	19	151	111	16	219	79	**0.997**	0.006
FA	TP	25	9,971	5,988	31	922	472	0.994	0.003
	TPS	736	**73,403**	48,820	78	14,422	6,569	0.987	0.008
	T	7	105	70	15	219	74	**0.985**	0.010
PA	TP	7	4,671	2,837	31	5,109	2,328	0.957	0.017
	TPS	141	18,058	10,547	78	**150,781**	74,342	0.921	0.026

3.3 Resulting Aggregation through Hierarchical Heavy Hitters

We have implemented the Full Ancestry and Partial Ancestry variants of the HHH algorithm mentioned in Section 3.1. The code was integrated into the RapidMiner data mining tool.[5]. Regarding run-time, all experiments were done on a machine with Intel Core 2 Duo E6300 processor with 2 GHz and 2 GiB main memory.

Since we want to aggregate system call data on devices that are severely limited in processing power and available memory, measuring the resource usage of our algorithms was of paramount importance. Table 20 shows the run-time and memory consumption of the Full Ancestry and Partial Ancestry algorithms using only the *system call type* hierarchy, the *system call type* and *file path* hierarchy, or the *system call type*, *file path*, and *call sequence* hierarchy. Minimum, maximum

[5] The code and all data which was used in the experiments is available at `http://www-ai.cs.uni-dortmund.de/SOFTWARE/HHHPlugin/`

and averages were calculated over a sample of the ten gathered log files for each of the eleven application by taking only the first log file for each application into account.

Memory consumption and run-time increase with the dimensionality of the elements, while at the same time approximation quality decreases. Quality is measured as similarity to the exact solution. Full Ancestry has a higher approximation quality in general. The results correspond to observations made by Cormode and are probably due to the fact that Partial Ancestry outputs bigger HHH sets, which was the case in our experiments, too. Note that approximation quality can always be increased by changing parameter ε to a smaller value at the expense of a longer run-time. Figure 6 shows the behaviour of our algorithms on the biggest log file (application Rhythmbox) for three dimensions with varying ε and constant ϕ. Memory consumption and quality decrease with increasing ε, while the run-time increases. So the most important trade-off involved here is weighting memory consumption against approximation quality — the run-time is only linearly affected by parameter ε. Again, Full Ancestry shows a better approximation quality in general.

Even for three-dimensional elements, memory consumption is quite low regarding the number of stored tuples. The largest number of tuples (73,403), only equates to a few hundred kilobytes in main memory! The longest run-time of 150,781 ms for Partial Ancestry in three dimensions relates to the size of the biggest log file (application Rhythmbox).

Classification Results. For the 110 log files of all applications, we determined the HHHs, resulting in sets of frequent tupels of hierarchical values. Interpreting each HHH set as an example of application behaviour, we wanted to answer the question if the profiles could be separated by a classifier. So we estimated the expected classification performance by a leave-one-out validation for kNN.

Therefore, we needed to define a distance measure for the profiles determined by HHH algorithms. The data structures of HHH algorithms contain a small subset of prefixes of stream elements.

The estimated frequencies f_p are calculated from such data structure by the output method and compared to ϕ, thereby generating a HHH set. The similarity measure DSM operates not on the HHH sets, but directly on the internal data structures D_1, D_2 of two HHH algorithms:

$$\mathrm{sim}(D_1, D_2) = \frac{\sum_{p \in P_1 \cap P_2} \mathrm{contrib}_{\mathrm{DSM}}(p)}{|P_1 \cup P_2|} .$$

Be f_p^i the estimated frequency of prefix p for data structure D_i as normally calculated by the HHH output method. The contribution of individual prefixes to overall similarity can then be defined as

$$\mathrm{contrib}_{\mathrm{DSM}}(p) = \frac{2 \cdot \min(f_p^1, f_p^2)}{\min(f_p^1, f_p^2) + \max(f_p^1, f_p^2)} .$$

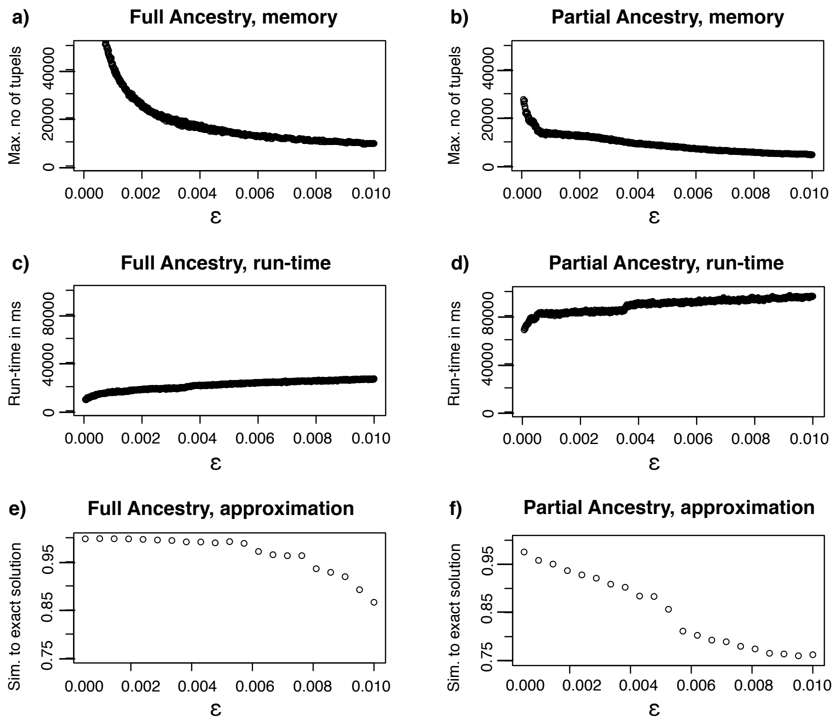


Fig. 6. Memory consumption (a, b), run-time (c, d) and similarity to exact solution (e, f) of HHH algorithms (three-dimensional) with varying ε, $\phi = 0.001$ on biggest log file of application Rhythmbox.

Table 21. Results for kNN (k = 3, 5, 7, 9), $\varepsilon = 0.0005, \phi = 0.002$ and distance measures DSM and TF, when only the *system call type* hierarchy or *system call type* and *call sequence* hierarchy together are used

	T		**TS**	
k	DSM	TF	DSM	TF
3	**10.3**	17.0	**7.7**	17.0
5	**12.7**	18.7	**8.7**	18.7
7	**14.0**	21.7	**8.7**	21.7
9	**14.0**	21.0	**9.0**	21.0

The so defined similarity measure is independent from the choice of ϕ, as no HHH sets need to be calculated in the time-consuming *Output* operation of the algorithms.

The classification errors for different values of k, hierarchies and distance measures are shown in Tab. 21. The new DSM distance measure which is independent of parameter ϕ shows the lowest classification error in all validation experiments.

As a baseline, we also determined the relative frequencies (TF, term frequencies) of call types per log file and classified them using kNN (with Euclidean distance). The error for profiling by HHH sets is significantly lower than for the baseline.

4 Conclusion

Server-based and device-based learning has been investigated regarding resource constraints. Further experiments to measure resource consumption will be conducted on real mobile devices, like Android mobile phones, whose operating system is also based on the Linux kernel.

Additionally, in order to estimate the *end-user* benefits of the approach in our server-based learning scenario, we have almost finished the development of a system simulator. The simulator consists of a precise disk model[6], a cache simulation with parameterizable cache size and page replacement strategy, a flexible I/O scheduler, a process execution simulator, and a plug-in interface for arbitrary prefetching strategies. It processes system-call traces and calculates total execution times as well as CPU and I/O subsystem loads. Based on this simulator we plan to study the implementation subtleties of prefetching strategies and to compare our learning-based prefetching with the latest heuristics of the Linux kernel.

Aggregation using HHH worked successfully for the classification of applications. Further work will exploit HHH aggregation for other learning tasks and inspect other data streaming algorithms. Concerning server-based learning, we may now answer the questions from the introduction, whether structural models are demanded for the prediction of user behavior on the basis of system calls, or simpler models such as Naive Bayes suffice. Should the sequential nature of system calls be taken into account by the algorithm? Or is it sufficient to encode the sequences into the features? Or should features as well as algorithm be capable of explicitly addressing sequences? We have compared CRF and NB with respect to their model quality, memory consumption, and runtime. Neither encoding the sequence into features nor applying an algorithm which is made for sequential information (i.e., CRF) outperforms a simple model (i.e., NB).

This is in contrast with studies on intrusion detection, where it was shown advantageous to take into account the structure of system calls, utilizing Conditional Random Fields (CRF) [11] and special kernel functions to measure the similarity of sequences [25]. Structured models in terms of special tree kernel functions outperformed n-gram representations when detecting malicious SQL queries [1]. Possibly, for prefetching strategies, the temporal order of system calls is not as important as we expected it to be. In the near future the resulting improvements in terms of cache hit rate and file operation latencies will be evaluated systematically based on a cache simulator and by modifying the Linux kernel.

Given regular processors, CRF are only applicable in server-based learning. Possibly, the integration of special processors into devices and a massively

[6] By integration of disksim [3].

parallel training algorithm could speed up CRF for device-based learning. Further work will implement CRF on a GPGPU (general purpose graphic processing unit). GPGPUs will soon be used by mobile devices. It has been shown that their energy efficiency is advantagous [26].

Acknowledgements. This work has been supported by the DFG, Collaborative Research Center SFB876, project A1.

References

1. Bockermann, C., Apel, M., Meier, M.: Learning sql for database intrusion detection using context-sensitive modelling. In: Proc. 6th Detection of Intrusions and Malware, and Vulnerability Assessment, pp. 196–205. Springer, Heidelberg (2009)
2. Bovet, D., Cesati, M.: Understanding the Linux Kernel, 3rd edn. O'Reilly & Associates, Inc., Sebastopol (2005)
3. Bucy, J.S., Schindler, J., Schlosser, S.W., Ganger, G.R.: The disksim simulation environment version 4.0 reference manual. Tech. Rep. CMU-PDL-08-101, Carnegie Mellon University (May 2008)
4. Cantrill, B.M., Shapiro, M.W., Leventhal, A.H.: Dynamic instrumentation of production systems. In: Proc. of USENIX ATEC 2004. USENIX, Berkeley (2004)
5. Cormode, G., Korn, F., Muthukrishnan, S., Srivastava, D.: Finding hierarchical heavy hitters in data streams. In: VLDB 2003: Proceedings of the 29th International Conference on Very Large Data Bases, pp. 464–475. VLDB Endowment (2003)
6. Cormode, G., Korn, F., Muthukrishnan, S., Srivastava, D.: Diamond in the rough: finding hierarchical heavy hitters in multi-dimensional data. In: SIGMOD 2004: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, pp. 155–166. ACM, New York (2004)
7. Cormode, G., Korn, F., Muthukrishnan, S., Srivastava, D.: Finding hierarchical heavy hitters in streaming data. ACM Trans. Knowl. Discov. Data 1(4), 1–48 (2008)
8. Domingos, P., Pazzani, M.: Beyond independence: Conditions for the optimality of the simple bayesian classifier. In: Machine Learning, pp. 105–112. Morgan Kaufmann, San Francisco (1996)
9. Eigler, F., Hat, R.: Problem solving with systemtap. In: Proceedings of the Ottawa Linux Symposium, vol. 2006 (2006)
10. Frank, A., Asuncion, A.: UCI machine learning repository (2010)
11. Gupta, K., Nath, B., Ramamohanarao, K.: Conditional random fields for intrusion detection. In: 21st Intl. Conf. on Adv. Information Netw. and Appl., pp. 203–208 (2007)
12. Hastie, T., Tibshirani, R., Friedman, J.H.: The Elements of Statistical Learning, corrected edn. Springer, Heidelberg (2003)
13. Huang, J., Lu, J., Ling, L.C.X.: Comparing naive bayes, decision trees, and svm with auc and accuracy. In: Third IEEE International Conference on Data Mining, ICDM 2003, pp. 553–556. IEEE Computer Society, Los Alamitos (2003)
14. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proc. 18th International Conf. on Machine Learning, pp. 282–289 (2001)

15. Lohmann, D., Hofer, W., Schröder-Preikschat, W., Streicher, J., Spinczyk, O.: CiAO: An aspect-oriented operating-system family for resource-constrained embedded systems. In: Proc. of USENIX ATEC. USENIX, Berkeley (2009)
16. Malouf, R.: A comparison of algorithms for maximum entropy parameter estimation. In: COLING-02: Proceedings of the 6th Conference on Natural Language Learning, pp. 1–7. Association for Computational Linguistics, Morristown (2002)
17. Nocedal, J.: Updating quasi-newton matrices with limited storage. Mathematics of Computation 35(151), 773–782 (1980)
18. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE 77(2), 257–286 (1989)
19. Schraudolph, N.N., Graepel, T.: Conjugate directions for stochastic gradient descent. In: Dorronsoro, J.R. (ed.) ICANN 2002. LNCS, vol. 2415, pp. 1351–1358. Springer, Heidelberg (2002)
20. Schraudolph, N.N., Yu, J., Günter, S.: A stochastic quasi-Newton method for online convex optimization. In: Meila, M., Shen, X. (eds.) Proc. 11th Intl. Conf. Artificial Intelligence and Statistics (AIstats). Workshop and Conference Proceedings, jmlr, San Juan, Puerto Rico, vol. 2, pp. 436–443 (2007)
21. Sha, F., Pereira, F.: Shallow parsing with conditional random fields. In: NAACL 2003: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, pp. 134–141. Association for Computational Linguistics, Morristown (2003)
22. Silberschatz, A., Galvin, P.B., Gagne, G.: Operating System Concepts. Wiley Publishing, Chichester (2010)
23. Sutton, C., McCallum, A.: An Introduction to Conditional Random Fields for Relational Learning. In: Getoor, L., Taskar, B. (eds.) Introduction to Statistical Relational Learning, MIT Press, Cambridge (2007)
24. Tartler, R., Lohmann, D., Schröder-Preikschat, W., Spinczyk, O.: Dynamic AspectC++: Generic advice at any time. In: The 8th Int. Conf. on Software Methodologies, Tools and Techniques, IOS Press, Prague (2009) (to appear)
25. Tian, S., Mu, S., Yin, C.: Sequence-similarity kernels for SVMs to detect anomalies in system calls. Neurocomput. 70(4-6), 859–866 (2007)
26. Timm, C., Gelenberg, A., Weichert, F., Marwedel, P.: Reducing the Energy Consumption of Embedded Systems by Integrating General Purpose GPUs. Tech. Rep. 829, Technische Universität Dortmund, Fakultät für Informatik (2010)
27. Vishwanathan, S.V.N., Schraudolph, N.N., Schmidt, M.W., Murphy, K.P.: Accelerated training of conditional random fields with stochastic gradient methods. In: ICML 2006: Proceedings of the 23rd International Conference on Machine Learning, pp. 969–976. ACM, New York (2006)

Bayesian Networks to Predict Data Mining Algorithm Behavior in Ubiquitous Computing Environments

Aysegul Cayci, Santiago Eibe, Ernestina Menasalvas, and Yucel Saygin[*]

Sabanci University, Istanbul, Turkey,
Facultad de Informatica, Universidad Politecnica, Madrid, Spain
aysegulcayci@su.sabanciuniv.edu,
{seibe,emenasalvas}@fi.upm.es,
ysaygin@sabanciuniv.edu

Abstract. The growing demand of data mining services for ubiquitous computing environments necessitates deployment of appropriate mechanisms that make use of circumstantial factors to adapt the data mining behavior. Despite the efforts and results so far for efficient parameter tuning, incorporating dynamically changing context information on the parameter setting decision is lacking in the present work. Thus, Bayesian networks are used to learn, in possible situations the effects of data mining algorithm parameters on the final model obtained. Based on this knowledge, we propose to infer future algorithm configurations appropriate for situations. Instantiation of the approach for association rules is also shown in the paper and the feasibility of the approach is validated by the experimentation.

Keywords: automatic data mining, data mining configuration, ubiquitous data mining.

1 Introduction

Ubiquitous computing, being an immature computing paradigm, brings new challenges to software designers and also to the designers of data mining software. In ubiquitous computing, processing takes place on the restricted resource devices that are embedded or spread in the environment and moreover the context in which the devices are used is subject to change. An important implication of ubiquitous computing is the lack of expert involvement on tuning the software where expert knowledge is most needed due to the scarcity of resources and the variability of the context. Consequently, automatic configuration of software by considering the changing context and constrained resources, is essential.

There is an increasing demand for intelligent applications on ubiquitous devices while data mining methods have been the main way to provide such intelligence. Nevertheless, important challenges need to be addressed on the ubiquitous

[*] This work has been partially financed by Spanish Ministry of Innovation under project TIN2008-05924.

data mining design, which two of them will be focused on in this paper. First of all, circumstantial factors such as the context and the resource limitations of the device should be considered when deciding how to configure the data mining algorithm. Secondly, there is a need to develop methods for the autonomous and adaptable execution of data mining. A number of context-aware and resource-aware data mining approaches have been proposed in the literature to deal with the issues posed due to ubiquity ([11] [12]). The proposed approaches consider the current context and/or resource availability to adjust parameters of data mining process or to determine the configuration setting of data mining in order to make autonomous decisions. However, in these approaches, knowledge from the past experiences is not incorporated into the decision mechanism of the parameter setting. As a result, settings are not adaptable in the sense that they do not improve over time based on past experience. A mechanism that adapts the data mining algorithm's configuration setting decisions according to the experiences learned, is lacking. In order to fulfill this deficiency, we propose to use machine learning techniques for deciding parameter settings in a given situation according to past behavior of the algorithm.

Automatic parameter tuning research area has gained much interest in the recent years. Several studies have been published offering optimization and machine learning techniques to solve the problem. The main idea behind the optimization techniques is to determine the performance criteria to be optimized and the configuration that best satisfies this criteria. Optimization methods proposed for automatic parameter tuning are as follows: racing algorithm by Birattari et al ([4]); iterated local search approach by Hutter, Hoos and Stutzle ([15]); algorithm portfolios paradigm by Gagliolo and Schmidhuber ([10]); experimental design combined with local search by Diaz and Laguna ([9]). Other prominent technique proposed for automatic parameter tuning is based on machine learning classifiers. In general terms, classifiers are used to learn the parameters to set the configuration. Srivastava and Mediratta ([20]) suggest usage of decision trees for automatic tuning of search algorithms. Through classification of previous runs of the algorithm by means of Bayesian network, Pavon, Diaz and Luzon ([18]) have automatized the parameter tuning process.

Interest on automatic parameter tuning originates in alleviating configuration setting of algorithms with a plethora of parameters most of the time but not to provide autonomy. In general, the argument of current work on automatic parameter setting is to find a configuration regardless of the circumstances. In the current proposed methods, neither the state of the device nor the requirements of the current situation are considered. However, in ubiquitous computing environments, state of the device and the current situation are important factors to determine the appropriate parameter settings and to make the device behave autonomously under any circumstance. In our mechanism, we consider the relationship between situations and parameters. Specifically, context in which the device is in when data mining request is received and the availability of the resources are incorporated in the parameter setting decision.

Cao, Gorodetsky and Mitkas ([6]) discuss the contribution of data mining to agent intelligence. They argue that a combination of autonomous agents with data mining supplied knowledge provides adaptability whereas knowledge acquisition with data mining for adaptability relies on past data (past decisions, actions, and so on). Our approach to provide adaptability is similar: we use machine learning approach in order to generate adaptable parameter setting decisions and enhance ubiquitous data mining with autonomy and adaptability. Our mechanism is based on Bayesian networks because Bayesian networks enable (1) the finding of the probabilistic relationships between the circumstances, parameters, and performance criteria, (2) considering several factors rather than a single criteria when determining the setting and (3) adaptability by learning from the past experiences. Pavon, Diaz, Laza and Luzon also proposed Bayesian networks for parameter tuning in [18]. The innovative feature of our mechanism is that we use not only information on parameters but also information on context and resources (what we call circumstances) to discover the appropriate configuration of a data mining algorithm for a given situation. When determining the best configuration, both efficiency of the data mining process and efficacy of the final model are taken into account.

The rest of the paper is organized as follows: Section 2, presents the proposed approach whereas section 3 instantiates it for a case. In Section 4, we explain the experiment that we performed in order to validate of the proposed approach and finally, section 5 is the conclusion.

2 Automatic Configuration for Ubiquitous Data Mining

We present a mechanism to predict the appropriate settings of a data mining algorithm's parameters in a resource-aware and context-aware manner. The mechanism is based on learning from past experiences, that is, learning from the past executions of the algorithm in order to improve the future decisions.

2.1 Analysis of the Problem

Our goal is to configure automatically a data mining algorithm which will run on a ubiquitous device. Since circumstantial factors such as the conditions of the resources and the context in which the device is used are important in a ubiquitous computing environment, availability of the knowledge on the following is useful for determining the algorithm's appropriate configuration:

– the resources that the algorithm needs in order to accomplish its task,
– the algorithm parameters that have an effect on the resource usage or on the data model quality,
– the context features which may have an effect on the efficacy of the data mining model or the efficiency of data mining,
– the features of the mining data set,
– the quality indicators which show the efficacy of the data mining model and efficiency of the data mining.

On the other hand, the problem that we tackle also implies the solution to address an important issue which is to change or improve the configuration setting decisions as the circumstances change. That means that, automatically generated configuration decisions must be adapted to the changing conditions just like a data miner expert who adapts his decisions when the conditions change.

Next, we define the factors for configuration that we derive from the items outlined above, and then present our solution to the problem after briefly introducing the Bayesian networks which we use in our proposed mechanism.

2.2 Problem Definition

When deciding how to set the parameters of an algorithm for a specific run, we determined that in a ubiquitous computing environment, circumstantial factors should be taken into account as well as the required quality. For this reason, we grouped the relevant factors for the configuration as circumstance and quality. Conditions of the device's resources and information on context belong to circumstance factor whereas level of efficiency required for the data mining process and data mining model expected quality level are quality factors. Formal definition of the algorithm configuration and relevant factors are as follows:

Circumstance: Circumstance, denoted by C, is defined by a set of ordered pairs *(f,s)* where f is either a resource or context feature and s is the state of this feature.

Quality: Quality, denoted by Q, is defined by a set of ordered pairs *(q,l)* where q is a quality feature and l is the required level for this quality. Quality features are metrics of efficiency or efficacy of the algorithm.

Parameters: Configuration of the algorithm, denoted by P, is defined by a set of ordered pairs *(p,v)* where p stands for a parameter and v is the value it takes.

In this way, we covered all but the "features of the mining data set" outlined in problem analysis (subsection 2.1). We deliberately disregarded the effect of mining data set features on the configuration decision for the time being because we want to focus on the ubiquitous aspect in this work.

Automatic configuration for ubiquitous data mining: Let Q be the required quality and C be the circumstance sensed or observed, then the configuration of data mining algorithm P is obtained by the mapping:

$$f : C \times Q \rightarrow P$$

We propose to use Bayesian networks to discover configuration of a data mining algorithm (P), aiming to attain the requested quality (Q), for the circumstance (C) observed when a data mining request is issued.

2.3 Bayesian Networks

Bayesian networks which represent the joint probability distributions for a set of domain variables are proved to be useful as a method of reasoning in several research areas. Medical diagnosis([3]), language understanding ([7]), network fault detection([14]) and ecology([2]) are just a few of the diverse number of application areas where Bayesian network modeling is exploited.

A Bayesian network is a directed acyclic graph that shows the conditional dependencies between domain variables and may also be used to illustrate graphically the probabilistic causal relationships among domain variables. The nodes of the network represent the domain variables and an arc between two nodes (parent and child) indicates the existence of dependency among these two nodes. Conditional probabilities of the dependencies among each variable and its parents are also represented along with Bayesian networks. The joint probability of instantiated n variables (i.e. variable x_i has an assigned value) in a Bayesian network is computed by:

$$P(x_1, ..., x_n) = \prod_{i=1}^{n} P(x_i | parents(X_i)) \tag{1}$$

where $parents(X_i)$ denotes the instantiated parents of the node of variable X_i.

Bayesian networks which are convenient for probabilistic inferencing, are commonly used for predicting the values of the subset of variables given the values of observed variables. In depth knowledge on Bayesian networks can be found in [19].

Learning the Bayesian network structure rather than creating the structure by analyzing the dependencies of domain variables, is a field of research which was studied extensively. Algorithms that learn the structure are most useful when there is a need to construct a complex network structure or when domain knowledge does not exist as in a ubiquitous computing environment. A discussion of the literature can be found in [5].

2.4 Behavior Model of the Data Mining Algorithm

In our solution, we propose to create a model that shows under possible circumstances what is the quality obtained against possible configurations of the data mining algorithm's execution. We call this model the *behavior model of the data mining algorithm* since the model represents the algorithm's behavior against different configurations and circumstances. We use machine learning as the main mechanism for creating the behavior model, in particular, we propose to construct a Bayesian network from *execution data* collected during algorithm's previous executions.

Execution Data. Execution data, denoted by E, consists of three types of attributes E=(C^E,Q^E,P^E) where the current circumstance just before the data mining algorithm runs, is stored in the set of attributes $C^E = \{c_1, c_2, ..., c_n\}$, the quality detected at the end of algorithm's execution in $Q^E = \{q_1, q_2, ..., q_m\}$, and finally, the configuration of the current execution in $P^E = \{p_1, p_2, ..., p_k\}$.

The attribute sets of execution data, C^E,Q^E, and P^E are supersets of the domain sets of every possible C, Q, and P given in problem definition (subsection 2.2). Let C^{dom},Q^{dom}, and P^{dom} be the domain sets of C, Q, and P respectively, then $C^{dom} \subseteq C^E$, $Q^{dom} \subseteq Q^E$, and $P^{dom} \subseteq P^E$. For example, assume $C = \{(f_1, s_1), (f_2, s_2)\}$, then $C^{dom} = \{f_1, f_2\}$ and $f_1, f_2 \in C^E$. This means that, information about all kinds of circumstances and quality requirements that may occur and therefore should be taken into account for the configuration decisions, are collected during algorithm's execution and stored in execution data. Similarly, automatic setting of every parameter value is captured.

Each execution of the data mining algorithm creates an instance in E and thus a history of executions for the data mining algorithm is formed which will be used to construct the behavior model.

Behavior Model. A Bayesian network is learned from the execution data and, afterwards, this Bayesian network representing the behavior model, is used to predict the appropriate configurations for the algorithm. Fig. 1 illustrates Bayesian network construction steps that we propose. K2 algorithm proposed by Cooper and Herskovits([8]) was used when constructing the Bayesian network. Initial step of behavior model generation is to discretize the execution data since K2 assumes database variables to be discrete. K2 learns Bayesian network structure from database of cases (E in our case) by determining the most probable network structure B_s given E:

$$\max_{B_s}[P(B_s|E)] \tag{2}$$

We made use of the open source code of Weka software([13]) to construct the network and updated it to fit our needs. The original algorithm seeks relationships among all the variables. However, execution data has three groups of variables (C^E,Q^E,P^E) and the relationships among the variables within a group such as the relationship among circumstantial variables *location* and *memory available* are not interesting. For this reason, modification of the K2 algorithm is necessary to look for relationships among nodes belonging to different groups of variables. A level (lvl) is assigned to each group based on the possible cause-effect relationship between them. The levels are used to prevent the nodes in the lower level groups to be the parents of the nodes in the upper level groups. Let $V = \{v_{l,k} | l = 1, ..., lvl \text{ and } k = 1, ..., x_l\}$ be the set of nodes of Bayesian network where x_l is the number of nodes in level l and $v_{i,j}, v_{m,n} \in V$. Then, i) nodes $v_{i,j}$ and $v_{m,n}$ can not be related if $i = m$, ii) $v_{i,j}$ can be the immediate parent node of $v_{m,n}$ only if $m = i + 1$.

The Bayesian network that is constructed from past execution data represents the probabilistic relationship between circumstance states, discretized possible parameter settings, and measured as well as discretized quality indicators. Appropriate setting of an algorithm's parameter is extracted from the Bayesian network as explained in the next section.

2.5 Mechanism to Predict Ubiquitous Data Mining Configuration

Once the behavior model is built, the steps that lead to automatic parameter configuration are as follows (See Fig. 1 for details):

- A data mining model is needed for a specific data set,
- Current circumstance (C) is observed and the quality requirements (Q) are acquired,
- Configuration (P) of the data mining algorithm is determined autonomously by inferencing from the behavior model

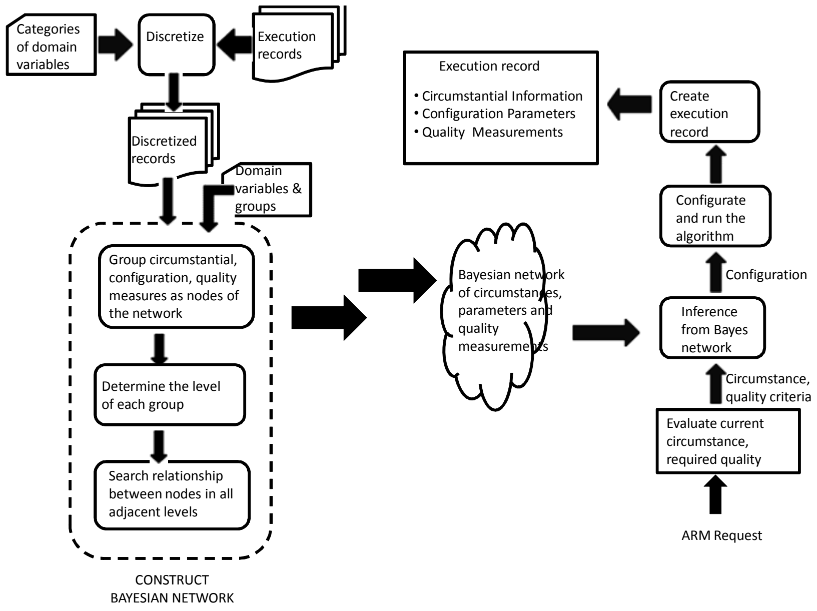

Categories of domain variables
Discretize
Execution records
Discretized records
Domain variables & groups
Group circumstantial, configuration, quality measures as nodes of the network
Determine the level of each group
Search relationship between nodes in all adjacent levels
CONSTRUCT BAYESIAN NETWORK
Bayesian network of circumstances, parameters and quality measurements
Execution record
• Circumstantial Information
• Configuration Parameters
• Quality Measurements
Create execution record
Configurate and run the algorithm
Configuration
Inference from Bayes network
Circumstance, quality criteria
Evaluate current circumstance, required quality
ARM Request

Fig. 1. Bayesian Network construction steps and data mining configuration mechanism

The most likely configuration is inferred from the behavior model by estimating the probabilities of possible parameter settings from previous runs of the algorithm in execution circumstances similar to current in order to obtain quality levels similar to the required. In particular, $p(x|F)$ which is the conditional probability of x (an instantiation of parameter variable) given F (instantiations of circumstance and quality variables), is evaluated. A pseudo code of this calculation is given below.

Pseudo code of parameter setting estimation from the Bayesian network

```
Definitions:
Let $C$ be the set of circumstance sets $C_k$
     where $(f_{kj},s_{kj})$ is a feature, state pair in $C_k$,
            $c_k$ is the number of pairs in $C_k$.
    $Q_k$ is the corresponding quality set of $C_k$
     where $(q_{kj},v_{kj})$ is a feature, state pair in $Q_k$,
            $n_k$ is the number of pairs in $Q_k$.
    $P$ is the set of parameters sets $P_x$
     where $p_{xy}$ is a parameter setting in $P_x$,
           $r_x$ is the number of possible settings for $P_x$.
Pseudo Code:
     for every $C_k$ in $C$
      let $F$ be all $f_{kj}=s_{kj}$ (forall j <= $c_k$) and $q_{kl}=v_{il}$ (forall l <= $n_k$)
          for every $P_x$ in $P$
              for every $p_{xy}$ in $P_x$ (forall y <= $r_x$)
                     calculate $Pr_{xy}$ = Probability ($P_x$ = $p_{xy}$ | $F$ )
              $p_{xy}$ with highest Pr is the appropriate setting of $P_x$ for $C_k$.
```

3 Autonomous Data Mining for Museum Guide

In order to increase the comprehensibility of our approach, we illustrate its use by instantiation. For this purpose, we describe an example application where ubiquitous devices are used by many people performing the same activity and somehow sharing information on their activity among themselves. In this sense, they form a social network in the ambient intelligence environment that we delineate. We explain how our approach can be employed in this application and finally instantiate our approach with data specific to this example application.

3.1 Motivating Example: A Museum Equipped with Ambient Intelligence

The museum depicted in this example is huge so that it takes a lot of time of the visitors to see all the exhibitions in it and it may even be impossible within the visit time. Museum is conceived as an ambient intelligence environment where visitors are fed information from the environment. The aim is to guide the visitors so that they can visit the pieces (art work) that they would like to see within the available time rather than trying to see all of the exhibitions. For this purpose, an application that we call *museum guide* is loaded to the smartphones of the visitors upon request. *museum guide* directs a visitor to the pieces that he would like in the museum.

As in a common web based book or movie recommender system, the objective of this system is to discover common likes of users. In order to do that, the amount of time each visitor spends in front of a piece is sensed and collected. The amount of time spent in front of a piece is taken as the indication of that the visitor's liking of that piece. Sensed data of each visitor is transformed to a record that we call *visitor record. visitor record* contains the pieces liked by that visitor.

museum guide which runs on the smartphone of a specific visitor while recording the pieces he liked to his smartphone, also downloads other *visitors' records.* Common likes are discovered by association rule mining of all *visitors' records* on the smartphones of the visitors. The purpose is to find the associations such as "visitors who liked Picasso's Three Musicians also liked Matisse's Dance". This museum has special offers for different types of visitors such as tourists, students and elderly. For this reason, visitor profile in one season, month or week of day may be quite different than the other. Since more associations can be found when similar people's likings are mined and the museum has dynamic visitor profile by day, rather than discovering the associations only once from data of one set of visitors, it is preferred to extract the association rules dynamically on visitors' smartphones. Moreover, continuous rotation of artwork also necessitates mining to be performed dynamically.

We focus on determining the configuration of the association rule mining algorithm which must run autonomously since in an ubiquitous computing environment it is assumed that the user can not provide this information. We exploit context and consider the availability of the resources of the visitor's smartphone as the determining factor of the association rule mining parameter settings. The data mining process is referred as *discover artwork associations.* Next, we describe a number of principles of *museum guide* and how it interacts with *discover artwork associations*:

- *museum guide* calls *discover artwork associations* to discover frequent itemsets from the set of *visitor records.* The attribute of each *visitor record* is the list of pieces liked by that visitor.
- *museum guide* downloads fresh data and calls *discover artwork associations* several times during his visit for a visitor in order to incorporate data of newcomers and to try different parameters for better recommendations.
- Association rule mining algorithm Apriori [1] which accepts two parameters: *minimum support* and *minimum confidence* is used to *discover artwork associations.*
- The resulting data mining model generated by Apriori are the association rules demonstrating which pieces that are exhibited in the museum are liked by the same persons. It is out of the scope of this work to speculate on how *museum guide* uses the model to recommend artwork to the visitor or whether the recommendations are ordered by physical location or some other criteria as well as when a new model is needed. On the other hand, we are concerned on the automatic configuration of *discover artwork associations.*
- Past executions of *discover artwork associations* are mined to discover the appropriate configuration that fulfils the required quality under a given circumstance. Initial past execution data is downloaded to the smartphone together with the application and is augmented by data collected during executions of Apriori locally on the visitor's smartphone.

3.2 Circumstantial Factors Effecting Parameter Setting

We argue that in an ubiquitous computing environment, in order to find appropriate settings of the data mining algorithm parameters, context from the environment as well as the conditions of the device's resources need to be utilized. Next, we define the relevant circumstantial factors for determining configuration of *discover artwork associations.*

Context: Context, that we assume have an effect on the required quality of the final model are:
- time left to the museum's closing (*remaining time to close*)
- time left to average visit time since the start of visit time (*remaining time to leave*)
- visitor's past attitude against the recommendations (*feedback*)
- number of visitors in the gallery entered (*no of visitors*)

Resources: Since *discover artwork associations* runs on the smartphones that are restricted resource devices, the resource usage of the data mining process need to be considered when setting the parameters of the data mining process. We assume memory and processor are the resources whose availability are critical for *discover artwork associations*:
- amount of memory available (*memory available*)
- processor idle percentage (*processor idle percent*)

3.3 Heuristics for Parameter Setting

In this subsection, we give example heuristic configuration decisions for *discover artwork associations.* We assume there exist default settings for each *discover artwork associations* parameter: *minimum support* and *minimum confidence* and in the configuration decision whether to increase or decrease the defaults of the relevant parameters depending on the circumstance is indicated. The following are some heuristics which may be used when configuring *discover artwork associations* autonomously where the reasoning explaining the heuristic follows it.

1. if *memory available* is low then increase the *minimum support* (with a higher *minimum support* value it is expected to decrease the size of the frequent itemsets and to optimize memory usage consequently),
2. if *remaining time to close* is small then increase both the *minimum confidence* and *minimum support* (since limited time is left, provide less rules with higher confidence so that the visitor will not miss the pieces that he would like most),
3. if *remaining time to leave* is small given that *memory available* is not low, decrease the *minimum support* (the objective is to make the average visit time longer by providing more pieces to the visitor that he would regret if he would leave without seeing them)

4. if *no of visitors* is high then decrease the *minimum support* but increase the *minimum confidence* (as the visitor may prefer to skip the pieces with a crowded audience in front of them, produce a list of high confidence containing sufficient number of pieces to bypass some of them)
5. if *feedback* is negative then decrease the *minimum support* (if the visitor is not satisfied with the previous recommendations then provide more accuracy)

Circumstance is the motive of each parameter setting but there is also an objective of each recommended setting which is given in parentheses after each heuristic. Objectives can be quantified by making use of the measurements obtained from the operating system of the device as well as the data mining model quality indicators.

Quality Measures: Quality measurements are the means to control whether the heuristic for a setting achieves the objective. The suggested quality measurements are as follows:
- maximum amount of memory used by *discover artwork associations* (*max memory usage*)
- number of association rules in the model (*no of rules discovered*)
- minimum confidence that an association rule in the model may have (*model min conf*)
- minimum support that association rules of the model should have (*model min support*)

Whether the objective of assigning certain value(s) to *discover artwork associations*'s parameter(s) is attained at a specific run of *discover artwork associations* can be assessed by checking the related quality measure(s) after the *discover artwork associations* runs with those settings. Similarly, the objective given is the required quality for a given circumstance.

3.4 Instantiation

In this subsection, automatic configuration setting is instantiated for the well known association rule mining algorithm Apriori. Instantiation is based on the intelligent museum example and consists of appropriate *discover artwork associations* configurations for the heuristic parameter setting decisions given in subsection 3.3 as well as the possible circumstances and quality determined in subsection 3.2 for the intelligent museum example.

Circumstance. It is assumed that context and resource availability values are discretized such that *'low'*, *'high'* and *'moderate'* categories are used for *memory available*, *'few'* and and *'many'* for *no of visitors*, *'not much'* and *'plenty'* are used for *remaining time to close* and *remaining time to leave* and finally, *'positive'* and *'negative'* are for *feedback*. Some possible instantiations of circumstances using discretized values are as follows:

C_1= {(memory available, 'low')}
C_2= {(remaining time to close, 'not much')}

C_3= {(memory available, 'high'),(remaining time to leave, 'not much')}
C_4= {(no of visitors, 'many')}
C_5= {(feedback, 'negative')}

Quality. Similarly, it is also assumed that quality measurement values are discretized such that *'low'*, *'high'* and *'moderate'* categories are used for *max memory usage*, *'few'* and and *'many'* for *no of rules discovered*, *'low'* and *'high'* are used for both *model min conf* and *model min support.* Some possible instantiations of circumstances using discretized values are as follows:
Q_1 ={(max memory usage, 'low')}
Q_2 ={(model min conf, 'high'), (model min support, 'high')}
Q_3 ={(no of rules discovered, 'many')}
Q_4 ={(model min conf, 'high'), (no of rules discovered, 'few')}
Q_5 ={(model min support, 'low')}

Algorithm Configuration. Assuming that the default values of *minimum support* and *minimum confidence* are 0.75 and 0.9 respectively, in each of the parameter setting below either one of them or both are altered to meet the parameter setting decisions of the example heuristics.
P_1 = {(minimum support, 0.9), (minimum confidence, 0.9)}
P_2 = {(minimum support, 0.85), (minimum confidence, 0.98)}
P_3 = {(minimum support, 0.75), (minimum confidence, 0.9)}
P_4 = {(minimum support, 0.5), (minimum confidence, 0.98)}
P_5 = {(minimum support, 0.70), (minimum confidence, 0.9)}

Configuration Decisions. Instantiation of configuration decisions are based on the instantiations of circumstance, quality criteria and algorithm configuration such that for each circumstance C_i given above, Q_i is the corresponding quality aimed for C_i and P_i is the appropriate configuration setting given C_i and Q_i. The following pseudo code generalizes the instantiation of configuration decisions for the heuristics of subsection 3.3:

```
forall i < 6
   if $C_i$ is sensed/gauged and
      $Q_i$ is the corresponding required quality
      then $P_i$ is an appropriate configuration.
```

4 Experimental Evaluation

We conducted an empirical study to demonstrate that parameter setting decisions by the proposed mechanism are appropriate in the sense that they are good at delivering the quality requested for the circumstances. To validate the proposed mechanism, we selected Apriori as the data mining algorithm to be configured and created its behavior model in Bayesian network representation to derive configuration decisions. Besides, we employed another approach for parameter setting, full factorial experiment design and compared the inferences made from the Bayesian network against the results of the full factorial experiment design. The experiment has the following steps:

1. Ubiquitous data mining simulator is developed to generate experiment data.
2. Bayesian network is constructed in order to discover the probabilistic relationships among parameters, circumstances and quality.
3. Multi-level full factorial design is used to find out the parameters that are effective on the quality under a given circumstance.
4. The results of the two methods are compared to assess the proposed mechanism.

Fig. 2 shows the interaction of the experiment steps and the ubiquitous data mining simulator.

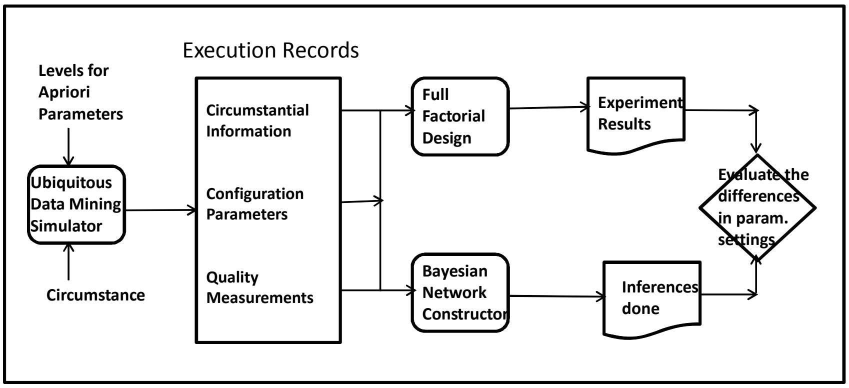

Execution Records
Levels for Apriori Parameters
Ubiquitous Data Mining Simulator
Circumstance
Circumstantial Information
Configuration Parameters
Quality Measurements
Full Factorial Design
Bayesian Network Constructor
Experiment Results
Inferences done
Evaluate the differences in param. settings

Fig. 2. Experiment phases

4.1 Ubiquitous Data Mining Simulator and Experiment Data

In order to collect data for automatic configuration of a specific data mining algorithm, we have developed a software to simulate the ubiquitous computing environment where data is mined by the same algorithm. Simulator runs the data mining algorithm with various configurations while generating the circumstances given and collects relevant execution data.

Simulator Architecture. Simulator runs Apriori which we selected for the experiments as the sample data mining algorithm to be configured, by calling Weka ([13]) API's. Since our purpose is to simulate the ubiquitous computing environment, Apriori was not executed when the device's resources are in arbitrary state. On the contrary, planned bottlenecks on the device's resources were created by the simulator before starting Apriori. Simulation software was implemented in JAVA language whereas bottleneck creator modules were written in C++ language. JAVA classes and their interactions are given in Figure 3.

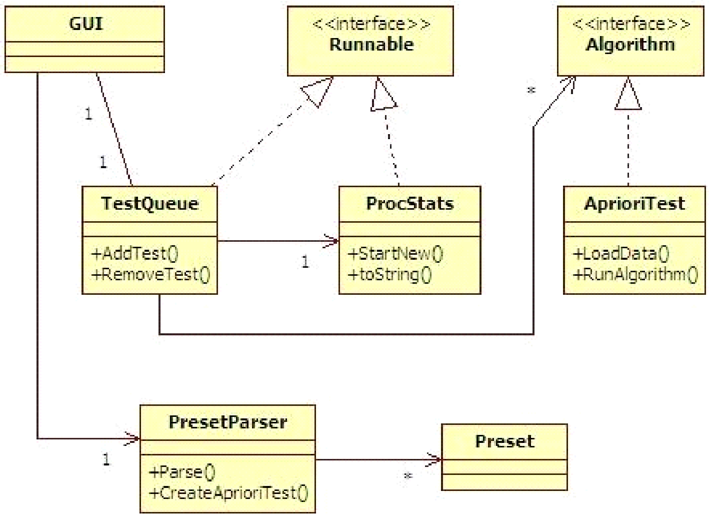


Fig. 3. Class descriptions of Ubiquitous Data Mining simulator

Input of the simulator is the preset file where each line represents a single test case. For each test case the following information are provided: context data associated with this Apriori run, resource bottleneck requests, data set to be mined, and parameter settings to be used for this test case. Resource bottleneck requests indicate the desired percent of memory and/or processor consumption by other workload in the device during execution of this test case.

Output of the simulator is the raw execution data in which there is a record for each execution of Apriori. Circumstance attributes (C^E) of execution data contain gauges showing resources' availability when Apriori was run and also the context which is given in the preset file for this execution. Data mining model as well as the measured, actual resource usage by Apriori are stored in quality attributes (Q^E). Together with parameter attributes (P^E), each execution record has 28 attributes.

Briefly, ubiquitous data mining simulator reads preset file, generates the resource scarcity conditions if the given circumstance requires and runs Apriori with the given parameters. Upon completion, an execution record is created. Ubiquitous data mining simulator (Figure 3) has a graphical interface (GUI) to set the name of the preset file and the execution file as well as to start the simulation. *PresetParser* is used to parse the contents of preset file and responsible for invoking bottleneck creators to call some "dummy programs" that will consume the requested amount of related resource. *TestQueue* is typically a queue that contains *Algorithm* instances. *AprioriTest* represents tests of the Apriori algorithm. and implements the interface *Algorithm*, thus its instances can be added to *TestQueue*. *ProcStats* performs the gathering of performance statistics before, after and during the execution of the algorithm tests. Specific system metrics related to memory or CPU are gathered using specific methods. This class is designed as an independent cohesive unit to measure performance metrics, gather system information and statistics.

Execution Data. Execution data for the experiment was generated using the ubiquitous data mining simulator. The states of the context and the type of resource constraints that we used in forming the circumstances of the test cases are {*home*, *office*} and {*short on memory*, *cpu bottleneck*, *none*} respectively. All the types of resource constraints were simulated for each context state, resulting in six different circumstances. Settings used for each Apriori parameters are given in Table 1. There are different sets of settings for *home* and *office*. In the experiment Apriori was run for all combinations of the determined settings for each of the six circumstance. Therefore, the number of test cases for each circumstance having *home* as context state are 2250(3x6x5x5x5) and *office* as context state is 180(3x3x5x2x2).

Table 1. Levels used for parameters

Context state	Mnemonic	Parameter	Settings
Home	U	upper bound minimum support	0.7, 0.8, 0.9
	M	lower bound minimum support	0.1, 0.2, 0.3, 0.4, 0.5, 0.6
	D	delta	0.01, 0.05, 0.1, 0.15, 0.2
	N	number of association rules	1, 5, 10, 15, 20
	C	minimum confidence	0.5, 0.6, 0.7, 0.8, 0.9
Office	U	upper bound minimum support	0.7, 0.8, 0.9
	M	lower bound minimum support	0.4, 0.5, 0.6
	D	delta	0.01, 0.05, 0.1, 0.15, 0.2
	N	number of association rules	15, 20
	C	minimum confidence	0.8, 0.9

4.2 Parameter Setting by Bayesian Network Inferences

In this step, we applied our mechanism to predict Apriori configurations from the Bayesian network. Bayesian network construction and inferencing from the network are two main tasks of this step.

Execution data generated by ubiquitous data mining simulator were first discretized before constructing the Bayesian network given in Fig. 4. While discretizing, we used equal frequency bins and chose the number of bins that produced the highest number of relationships the among nodes. While constructing the network we made use of the K2 algorithm ([8]) by modifying it to group the nodes and searched causal relationship among these groups of nodes. The nodes in the upper level of the network in Fig. 4 represent the circumstance, middle level nodes represent Apriori parameters, and finally the lowest level nodes are quality measures. The cause and effect relationships between circumstances and parameters present which parameter settings are appropriate under which circumstances, whereas the cause and effect relationships between parameters and quality measures show which parameters are effective on which quality measures.

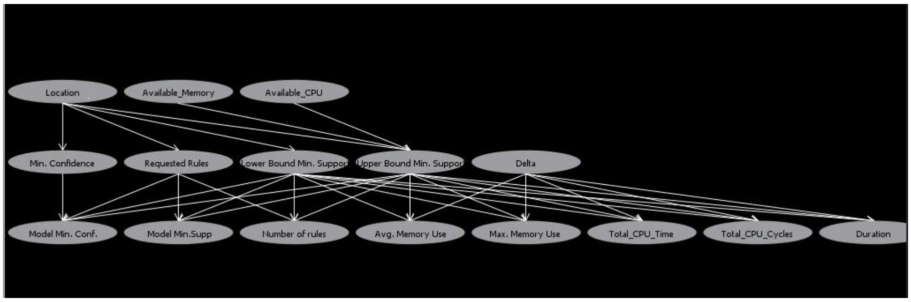


Fig. 4. Bayesian network of Apriori runs

While producing the experiment data for this Bayesian network, we did not determine appropriate parameter settings for circumstances but we ran Apriori for every combination of parameters in each circumstance because our purpose is to find the effect of parameters to quality measurements in the first place. Therefore, at this stage the relationships between circumstances and parameters are not meaningful. We assumed each circumstance node relates to each parameter node in order to include circumstances in the inference mechanism. The relationships between the parameter nodes and quality measure nodes represent the effectiveness of parameters against quality measurements. The Bayesian network in Fig. 4 shows that *minimum confidence* and *requested rules* are related only to efficacy; *delta* to all efficiency measurements as well as *lower and upper bound minimum support*, are related to all.

We determined parameter settings decisions by inferencing from the Bayesian network given in Fig. 4. The pseudocode of the estimation is given in subsection 2.5.

4.3 Multi-level Full-Factorial Experiment Design

In this step, we applied multi-level full factorial experiment design which is one of the Design of Experiment (DoE) methods [17]. Full factorial experiment design is statistically determining the effects of the factors of a process to its response by systematically varying the levels of the factors during testing of the process. In DoE terminology, *response* is the output variable of the process, *factors* are its input variables and *level* is a possible setting for a factor. The process that we want to analyze is the behavior of Apriori, more specifically, to find out which Apriori parameters affect which quality measurements. Therefore, Apriori parameters are the *factors*, their possible settings are the *levels* and resulting quality is the *response*. In full factorial experiment design, data is collected by running the process with all combinations of determined levels of its factors. Hence, we generated execution data similarly by running Apriori for all combinations of settings as explained in subsection 4.1. Moreover, since we ran Apriori by simulating six specific circumstances, we are able to analyze the effects for each circumstance.

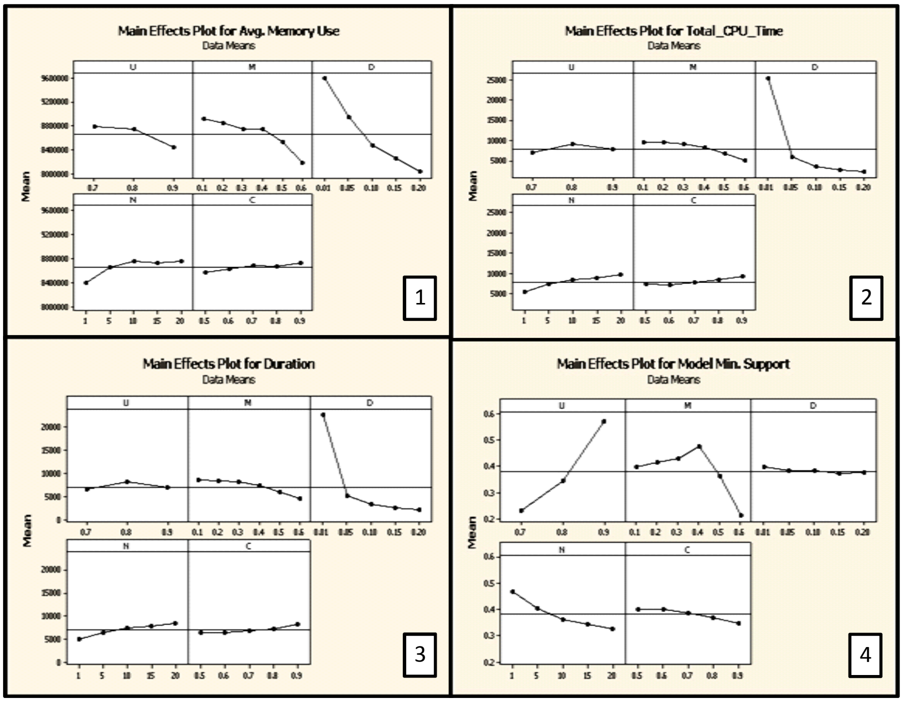


Fig. 5. Main effects plot of 4 quality measurements for *home-short on memory*

We used experiment software Minitab([16]) to estimate the effects and to plot the analysis results. Fig. 5 illustrates the full factorial experiment design results obtained for *home-memory low*. We analyze the results for this circumstance in detail in order to explain the method. In the figure, the means of quality measurements for the utilized levels of parameters are plotted. In quadrants of Fig. 5, plots for *average memory use*, *total CPU time*, *duration* and *minimum support of the model* are given respectively. Each plot (U, M, D, N, C) within a quadrant is for a parameter. The mean of the measured value is plotted for every level we tested for that parameter in the experiment. If the plot is not flat which indicates the means of measured values vary with different value assignments of this parameter, then this parameter is effective on the measured value. We considered the F test values to determine the significance of the effect. While determining the appropriate value of the parameter which is designated as effective on the measured criteria, we have chosen the value that has the smallest mean of response for its factor level combinations. We compare the results of full factorial experiment design against the results of the Bayesian network in the next subsection.

4.4 Comparison of Results

In order to determine the parameter settings of an algorithm, we explained two different approaches, Bayesian networks and full factorial experiment design where the former is a probabilistic approach and the latter a statistical approach. The outcomes of the approaches are summarized as follows:

- Full factorial design provides
  - The list of parameters which are not effective on a quality measure
  - The parameter setting which has the highest/lowest least square mean for a quality measure
- Inference from Bayesian network provides
  - The list of parameters which are not related to a quality measure
  - Most likely parameter setting given the circumstance(s) and the quality measure(s) as evidence

To compare the results, we used two criteria: i) the percentage of alike parameter/quality measure relationships and, ii) the percentage of identical parameter settings, obtained by the two approaches. In Table 2, for each circumstance, we present, (i) and (ii) by grouping quality measures as efficiency related and efficacy related.

Table 2. Comparison of results

Circumstance	Efficiency		Efficacy	
	(i)	(ii)	(i)	(ii)
home-short on memory	73	77	90	100
home-CPU bottleneck	80	89	100	100
home-no constraints	73	77	80	80
office-short on memory	100	67	100	75
office-CPU bottleneck	100	89	90	75
office-no constraints	100	78	90	75

It is possible to say based on the results (Table 2) that in majority of the cases, parameters that are found to be effective on a quality measure under a circumstance in full factorial design, are represented as related to that quality measure under the same circumstance in the Bayesian network. The appropriate parameter settings decided in order to optimize a quality measure in full factorial design is identical in most of the cases to the parameter settings inferred from the Bayesian network given the same quality measure.

4.5 Effects of Mining Data Set Feature Variations on the Behavior Model

In the simulation phase of the experiment (subsection 4.1), we mined always the same data set with Apriori and in this way we eliminated the effects of the data

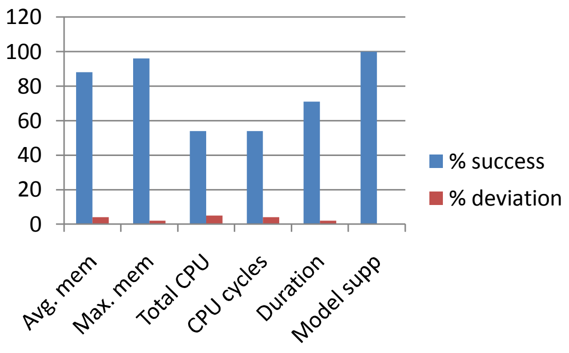


Fig. 6. Percentage of successful recommendations

set feature changes on the behavior model constructed. On the other hand, in a real life situation data set to be mined may grow or shrink either by addition or deletion of instances into the data set or by attribute set changes. Variations on the mining data set features may necessitate refreshing the behavior model that is used to recommend algorithm configurations for mining this data set. Thus, we performed a series of experiments to affirm that mining data set size may have an effect on the parameter setting recommendations and also to speculate on how to detect that the behavior model is decayed. In the experiments, we made use of quality measurement figures collected during the execution of Apriori to assess whether the recommended parameter settings provide the requested quality. Experiments rely on the behavior model that we call *basis behavior model* generated in the same way explained in subsection 4.2 from the execution records collected by running Apriori with input data set (*DSx1*) in a simulated ubiquitous computing environment similar to the one explained in subsection 4.1. Brief explanation of data set size variations effect evaluation experiments are as follows:

– **Verify the recommendations.** In this experiment, Apriori was configured by the recommended settings acquired from the *basis behavior model* and ran with input *DSx1* in the simulated ubiquitous computing environment for every possible recommendation. Afterwards, we determined the appropriateness of each recommendation by comparing the relevant quality measurement value collected during Apriori's execution against the requested quality used when deriving the recommendation from Bayesian network. For example, if an Apriori configuration is recommended to minimize the memory usage of Apriori, we assess the parameter setting objective by comparing the memory usage figures of Apriori's execution with this configuration against the lowest memory usage figures in the behavior model. The percentage of the Apriori executions which achieve the objective of the parameter settings grouped by relevant quality measurement are given in Figure 6. Percentage of deviation from the requested quality is also analyzed for each quality measurement

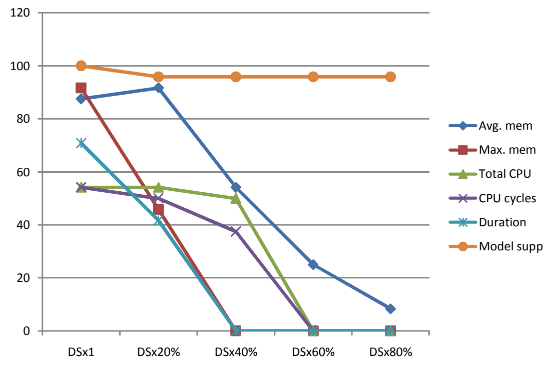


Fig. 7. Behavior model decay

group. The maximum amount of deviation is ten percent of the requested quality whereas the average amount of deviation does not exceed five percent of the requested quality for any of the groups (Figure 6). The results obtained are satisfactory to verify the appropriateness of the recommendations.

– **Demonstrate the data set size effect.** We try to find out in this experiment whether the behavior models extracted from the executions of the same data mining algorithm with same configuration settings but with different data mining data set sizes, are different. For this purpose, we generated another behavior model, *behavior model 10* in a similar way that we generated *basis behavior model* but the size of the data set (*DSx10*) used as input to Apriori in this experiment is ten fold bigger than *DSx1*. After generating the behavior model (*behavior model 10*) for Apriori mining *DSx10*, we compared *behavior model 10* against *basis behavior model* and detected that half of the recommendation decisions are changed. By this way, we have shown that input data set's size of a data mining algorithm may have an impact on certain parameter settings decisions given in order to achieve certain quality objectives.

– **Estimate behavior model decay.** In the final experiment, we gradually increased the size of the mining data set mimicking a possible real life situation in which a data set grows in time. Our purpose is to analyze the deterioration of the recommendations in terms of achieving the quality requested as the data set grows. We iteratively increased the size of the mining data set by twenty percent, run Apriori with all the possible recommended parameter settings extracted from the *basis behavior model* while simulating the relevant circumstance, recorded the requested quality for the recommendation and compare it against the achieved quality. During this process we used the same behavior model (*basis behavior model*) without populating new execution data or refreshing it completely. Figure 7 shows for different mining data set sizes the percentage of Apriori executions where the objective of the parameter setting is achieved in terms of the quality obtained. Experiment results show that the correctness of the configuration decisions derived from the behavior model in order to obtain the requested qualities of all types except the model minimum support are affected by the data set size change. Furthermore, *basis behavior model* decays needing a refresh before *DSx1* grows by forty percent. This experiment revealed that mining data set size change do not effect every parameter setting decision but if a parameter setting decision do not provide the requested quality, it is possible to detect.

5 Conclusion

Anticipation of the importance of autonomous and adaptable behavior in ubiquitous data mining, led us to research on how to tune its parameters automatically. In order to determine its appropriate configuration, we employed a mechanism to understand the data mining algorithm behavior. Considering the characteristics of ubiquitous computing environments, we stressed making use of circumstantial factors for determining the appropriate parameter values. We also aimed at recommending parameter settings that most likely satisfy the required quality of a circumstance. Thus, we analyzed the effects of parameter settings to quality measures which are related to both efficiency of data mining process and efficacy of the data mining model. We proposed to use Bayesian network for extracting data mining algorithm behavior while taking into account circumstance and the quality.

As the result of our simulation experiment, satisfactory parameter and quality measure relationships to recommend parameter settings, were formed in the Bayesian network. We also validated our proposal by comparing the parameter settings obtained from the Bayesian network against another approach, full factorial experiment design. Experiment on association rule mining shows that proposed method gives parameter settings almost identical to the settings obtained from full factor analysis which is a completely different approach. We also verified the correctness of the configuration recommendations derived from the behavior model by mining data with the recommended settings and obtained high percentage of correct configuration recommendations which produced mining results with required quality.

Experiment that we performed to show the negative effect of mining data set size change on the accuracy of the parameter setting decisions derived from the behavior model, revealed that behavior model may not be generated only once and reused forever but needs to be updated or refreshed with new experiences gained by the execution of data mining algorithm. Hence, health checking the behavior model is an important factor for successful automatic parameter tuning.

In the future, we aim to represent the mining data set characteristics such as the number of attributes and the number of tuples that are relevant for parameter tuning in the behavior model so that a single behavior model can be used for a wider set of configuration requests.

Acknowledgments. Authors would like to thank Can Tunca and Engin Dogusay from Sabanci University who contributed to this study by developing the supporting software.

References

1. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. In: Proceedings of the Int. Conf. on Very Large Data Bases (VLDB 1994), pp. 487–499. Morgan Kaufmann, San Francisco (1994)
2. Amstrup, S.C., Marcot, B.G., Douglas, D.C.: A Bayesian Network Modeling Approach to Forecasting the 21st Century Worldwide Status of Polar Bears. Arctic Sea Ice Decline: Observations, Projections, Mechanisms, and Implications. Geophysical Monograph 180, 487–499 (2008); American Geophysical Union, Washington, DC
3. Beinlich, I.A., Suermondt, H.J., Chavez, R.M., Cooper, G.E.: The ALARM Monitoring System: A Case Study with two Probabilistic Inference Techniques for Belief Networks. In: Proceedings of the Second European Conference on Artificial Intelligence in Medicine, London, pp. 247–256 (1989)
4. Birattari, M., Stutzle, T., Paquete, L., Varrentrapp, K.: A Racing Algorithm for Configuring Metaheuristics. In: GECCO 2002 Proceedings of the Genetic and Evolutionary Computation Conf., pp. 11–18. Morgan Kaufmann, San Francisco (2002)
5. Buntine, W.: A Guide to the Literature on Learning Probabilistic Networks from Data. IEEE Trans. on Knowl. and Data Eng. 8, 195–210 (1996)
6. Cao, L., Gorodetsky, V., Mitkas, P.A.: Agent Mining: The Synergy of Agents and Data Mining. IEEE Intelligent Systems 24, 64–72 (2009)
7. Charniak, E., Goldman, R.: A Semantics for Probabilistic Quantifier–Free First–Order Languages with Particular Application to Story Understanding. In: Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, Menlo Park, California, pp. 1074–1079 (1989)
8. Cooper, G.F., Herskovits, E.: A Bayesian Method for Constructing Bayesian Belief Networks from Databases. In: Seventh Conference on Uncertainty in Artificial Intelligence, pp. 86–94. Morgan Kaufmann, San Francisco (1991)
9. Adenso-Diaz, B., Laguna, M.: Fine-Tuning of Algorithms Using Fractional Experimental Designs and Local Search. Oper. Res. 54, 99–114 (2006)
10. Gagliolo, M., Schmidhuber, J.: Learning Dynamic Algorithm Portfolios. Annals of Mathematics and Artificial Intelligence 47, 295–328 (2006)
11. Gaber, M.M., Yu, P.S.: A Framework for Resource-Aware Knowledge Discovery in Data Streams: a Holistic Approach with its Application to Clustering. In: ACM Symposium on Applied Computing, pp. 649–656. ACM, New York (2006)
12. Haghighi, P.D., Zaslavsky, A., Krishnaswamy, S., Gaber, M.M.: Mobile Data Mining for Intelligent Healthcare Support. In: 42nd Hawaii international Conference on System Sciences, pp. 1–10. IEEE Computer Society, Washington, DC (2009)
13. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA Data Mining Software: An Update. SIGKDD Explorations 11 (2009)
14. Hood, A.C., Ji, C.: Proactive Network Fault Detection. In: Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution, INFOCOM, p. 1147. IEEE Computer Society, Washington, DC (1997)

15. Hutter, F., Hoos, H.H., Stutzle, T.: Automatic Algorithm Configuration Based on Local Search. In: 22nd National Conference on Artificial Intelligence, pp. 1152–1157. AAAI Press, Menlo Park (2007)
16. Minitab Inc., http://www.minitab.com/en-US/
17. Montgomery, D.C.: Design and Analysis of Experiments. John Wiley and Sons, Chichester (2006)
18. Pavon, R., Diaz, F., Laza, R., Luzon, V.: Automatic Parameter Tuning with a Bayesian Case-Based Reasoning System. A Case of Study. Expert Syst. Appl. 36, 3407–3420 (2009)
19. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco (1988)
20. Srivastava, B., Mediratta, A.: Domain-Dependent Parameter Selection of Search-Based Algorithms Compatible with User Performance Criteria. In: 20th National Conference on Artificial Intelligence, pp. 1386–1391. AAAI Press, Menlo Park (2005)

Online and Offline Trend Cluster Discovery in Spatially Distributed Data Streams

Anna Ciampi, Annalisa Appice, and Donato Malerba

Dipartimento di Informatica, Università degli Studi di Bari Aldo Moro
via Orabona, 4 - 70126 Bari - Italy
{aciampi,appice,malerba}@di.uniba.it

Abstract. Emerging real life applications, such as environmental compliance, ecological studies and meteorology, are characterized by real-time data acquisition through remote sensor networks. The most important aspect of the sensor readings is that they comprise a space dimension and a time dimension which are both information bearing. Additionally, they usually arrive at a rapid rate in a continuous, unbounded stream. Streaming prevents us from storing all readings and performing multiple scans of the entire data set. The drift of data distribution poses the additional problem of mining patterns which may change over the time. We address these challenges for the trend cluster cluster discovery, that is, the discovery of clusters of spatially close sensors which transmit readings, whose temporal variation, called trend polyline, is similar along the time horizon of a window. We present a stream framework which segments the stream into equally-sized windows, computes online intra-window trend clusters and stores these trend clusters in a database. Trend clusters are queried offline at any time, to determine trend clusters along larger windows (i.e. windows of windows). Experiments with several streams demonstrate the effectiveness of the proposed framework in discovering accurate and relevant to human trend clusters.

1 Introduction

Spatio-temporal data mining has recently attracted considerable attention in research and practitioner communities. The main reason for this interest is that datasets containing prominent spatial and temporal data elements are becoming ubiquitous. Despite advances made in spatio-temporal data mining, many spatio-temporal data analysis approaches take only a static view of a geospatial phenomenon [7]. These approaches extract a finite set of data points based on user-provided criteria (e.g. the region boundary and/or the time horizon of interest) and address data mining tasks for static data only. However, a static perspective is inadequate as data may arrive dynamically, continuously and with a drift of the underlying data distribution. On the other hand, a dynamic perspective is not free of charge, it poses new challenges such as avoiding multiple scans of the entire data sets, optimizing memory usage and mining drifting patterns.

In this work, we consider a spatially distributed and time dependent scenario, where sets of readings (snapshots) of a numeric dimension (measure) are streamed at time points, equally spaced in time, from a network of remote sensors. Both the spatial arrangement of sensors and the temporal dependence of consecutive readings are information bearing in this scenario. Sensors are spatially referenced on the Earth (e.g. by latitude and longitude) and the spatial location of each sensor in the network introduces a dependence among readings collected into a neighborhood. This spatial dependence, also observed in environmental and ecological domains, is known as Tobler's first law of geography [21], according to which "*everything is related to everything else, but near things are more related than distant ones*". Moreover, sensor readings, which are timestamped, may evolve in time. Thus, there is a temporal data component, referred to as *temporal locality* [2], which causes that data distribution is not static in time, but possibly subjected to concept drift.

Given these considerations, we capitalize on both the spatial arrangement of sensors and the temporal dependence of readings in order to address the task of computing trend clusters in a spatially distributed data stream. A trend cluster is a recently defined space-time cluster [9] which groups spatially close sensors whose readings vary according to a similar trend polyline along a time horizon in the stream. The trend polyline associated to the cluster is the sequence of straight-line segments which fit the clustered readings along the time horizon under consideration. This choice of the trend as a base pattern for clustering is motivated by the fact that knowing how "spatially clustered" readings evolve in time help us draw useful conclusions. For example, in a weather system it is useful to know how temperature increases and/or decreases over regions of the Earth and how the boundary of these regions changes or remains stable in time. On the other hand, a trend-based visualization of sensor readings reduces the amount of data , that is, only a trend polyline is plotted for each cluster of sensors, and is closer to the human interpretation. Finally, trend clusters are a compact representation (summary) of the stream to store in a database in place of the original data.

The innovative contribution of this work consists in the definition of a stream management framework, called TRUST (TRend clUsters from Spatio-Temporal data streams), which is able to discover trend clusters, both online and offline, with respect to the streaming environment of a sensor network. The online component operates continuously with the stream by completing the discovery process in real-time, discarding processed readings and storing discovered trend clusters into a database. The offline component is triggered from the user(s) request and it performs the discovery process by processing the patterns pasty stored in a database by the online component. For the online discovery, TRUST integrates SUMATRA (SUMmArization by TRend cluster discovery Algorithm), which segments the stream into equally sized windows of sensor readings. Once a window is completed in the stream, the online computation of the trend clusters along the window is triggered. For the scope of the offline discovery, TRUST integrates METRE (Merging TRend ClustErs), which retrieves trend cluster sets stored window-by-window in the database and use them to compute trend clusters which arise along the time horizon covering several consecutive windows (higher order windows). This inter-window discovery is performed offline with respect to the streaming activity, as previous readings are no longer available for further analysis.

The paper is organized as follows. In the next Section we report basic concepts and define trend clusters. Section 3 revises the related work on clustering in data streams and spatio-temporal data. Section 4 presents the stream management framework TRUST. Section 5 illustrates an experimental study of both online and offline trend cluster discovery techniques. Finally, conclusions are drawn.

2 Basic Concepts and Problem Definition

We introduce the basic concepts of spatially distributed data streams and we report a formal definition of the trend clusters in data streams.

2.1 Spatially Distributed Data Streams

We consider spatially distributed streams generated by the readings of a numeric measure which are continuously transmitted, equally spaced in time, by a network of remote (wireless) sensors. 2D points (e.g. latitude-longitude points) represent the spatial location of each networked sensor on the Earth. We assume that a point position can be occupied by a single sensor and the point position of each sensor is known, distinct and invariant (i.e. the sensor is not moving through the space). The number of transmitting sensors may change in time as a sensor may be inactive and transmits no reading for a time interval. Thus we are also able to face the case in which sensors are added to or definitively removed from the network at a time point.

Based upon these premises, the sensor readings of a spatially distributed data stream D are modeled by resorting to the *snapshot* data model originally defined in [3]. According to this model, the transmission time domain $\mathbb{T}$ is an unbounded series of discrete time points equally spaced in time. For each time point $t_i \in \mathbb{T}$, the snapshot D_i is the set of the sensor readings timestamped at t_i. The spatial arrangement of the readings within D_i is modeled by means of the field function [20] $f_i : K_i \mapsto \mathit{Attribute\ domain}$ where the field domain K_i is the set of point positions of remote sensors ($K_i \subseteq \mathbb{R}^2$) which transmit a reading at t_i. Though finite, K_i may vary with t_i as the number of transmitting sensors may change in time within the network. Based on this snapshot modeling of the stream, D is seen as a stream of snapshots $D_1, D_2, \ldots D_i, \ldots$.

The *count-based window model* [10] decomposes D into consecutive windows of w snapshots arriving in a series. w is the window size of the window model. Consecutive windows are progressively enumerated (first window, second window, ...) such that the i-th window comprises the w snapshots timestamped along the time horizon $[t_{(i-1)w+1}, t_{iw}]$. Thus snapshots are locally enumerated within the owner window (first snapshot, second snapshot and so on) and D is also modeled as a stream of consecutive windows of consecutive snapshots. Formally,

$$\begin{array}{c} D_1, \ldots, D_w\ , D_{w+1}, \ldots, D_{2w}\ ,\ \ldots\ , D_{(i-1)w+1}, \ldots, D_{iw}\ ,\ \ldots \\ \equiv \\ \underbrace{D_1^1, \ldots, D_w^1}_{W_1},\ \underbrace{D_1^2, \ldots, D_w^2}_{W_2}\ ,\ \ldots\ ,\ \underbrace{D_1^w, \ldots, D_w^w}_{W_i}\ ,\ \ldots \end{array} \quad (1)$$

where $D_{(i-1)w+j} \equiv D_j^i$, i is the index of the window in the stream and j is the local index of the snapshot in the window. j ranges between 1 and w.

2.2 Trend Clusters

A trend cluster (see Figure 2) is a triple defined as:

$$[\mathit{time\ horizon}, \mathit{spatial\ cluster}, \mathit{trend\ polyline}],$$

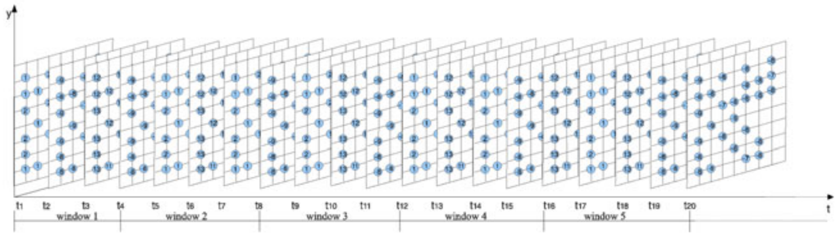


Fig. 1. Count-based window model of a stream of snapshots (window size $w = 4$). Each snapshot is mapped into a 2D Euclidean space, where sensors are drawn as blue circles. The number reported within a circle is the reading of the associated sensor at the time point of the snapshot.

where the $time\ horizon$ is a time interval, the $spatial\ cluster$ is the enumeration of spatially close grouped sensors and the $trend\ polyline$ is a piecewise line which compactly represents the trend variation of the clustered sensors along the time horizon. As we consider the count-based window, the time horizon is that of the window and it is implicitly defined by the window enumerative once the window size w is apriori known.

3 Related Works

In order to clarify the background of this work, related research on clustering in data stream mining and spatio-temporal data mining is reported below.

3.1 Data Stream Mining

In the last decade, several clustering techniques have been investigated in stream mining. Firstly Guha et al. [12] presented a constant factor of approximation for the k-Medians technique, which performs a single pass clustering in a data stream. Babcock et al. [4] extended this technique by framing the k-Medians technique into a window model. O'Callaghan et al. [19] defined a technique to cluster windows of incoming data and then re-group the clusters locally generated for each window. The limitation of both techniques is that their results are spherical clusters. They do not consider that clusters in data streams could be of arbitrary shape and number.

Aggarwal et al. [1] proposed a two-phase clustering technique, called CluStream, which separates out the clustering process into an online micro-clustering phase and an offline macro-clustering phase. The online micro-clustering phase stores the summary statistics (micro-clusters) of a fast data stream. The offline phase processes the summary statistics and provides an understanding of the clusters whenever required. Since the offline phase processes only summary statistics over a user-defined time horizon as input, it turns out to be very efficient in practice. This two-phased approach also provides the user with the flexibility to explore the evolution of the clusters over different time periods. The problem with CluStream is the predefined constant number of computed micro-clusters. Additionally, since a variant of k-means is adopted to obtain the final macro-clusters, a "natural" macro-cluster may be split into two parts.

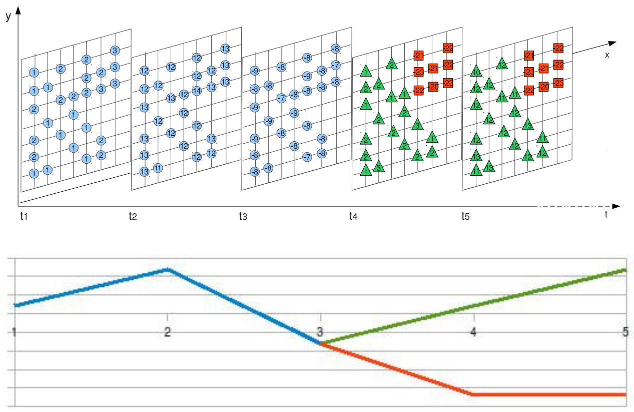


Fig. 2. Trend clusters ($w = 4$). The blue cluster groups circle sensors whose values variate as the blue polyline from t1 to t4. The red cluster groups squared sensors whose values variate as the red polyline from t5 to t8. The green cluster groups triangular sensors whose values variate as the green colored polyline from t5 to t8.

To discover arbitrary shaped clusters and handle outliers, Cao et al [6] proposed a density-based clustering technique, called DenStream, whose online component collects micro-clusters of the stream and associates each micro-cluster with a weight. This weight decays over time if no incoming data point is recently added to the micro-cluster. The offline component generates macro-clusters on demand from these micro-clusters by resorting to a variant of DBSCAN tailored to maintain online micro-clusters. In this phase, micro-clusters are regarded as virtual weight points located in the space. Virtual points whose weight is less than a threshold are outliers. As DenStream is based on DBSCAN, it is able to detect arbitrarily shaped clusters. Other density-based clustering techniques designed for data streams have been presented in [8,23].

It is noteworthy that no technique reported above takes into account the possibly distributed arrangement of readings. The exceptions are the naive-Bayesian network techniques [17,5] which permit to discover clusters from distributed data streams. Although these works are close to our research, the kind of pattern we discover, clusters with a trend polyline to describe how readings vary in time, requires a time-series processing of the readings which is not performed by these naive Bayesian network techniques.

3.2 Spatio-temporal Data Mining

Research in spatio-temporal data clustering focuses on the discovery of trajectory clusters and/or moving clusters in static data. Trajectory clusters group trajectories of a similar shape. This definition of a trajectory cluster motivates our consideration of the one-dimensional version of the trajectory clustering problem as the equivalent of the clustering of time-series that exhibit similar movements. Vlachos et al. [22] first defined a Least Common Subsequence distance, which is used as a base to apply any traditional clustering technique to object trajectories. Gaffney and Smyth [11] proposed

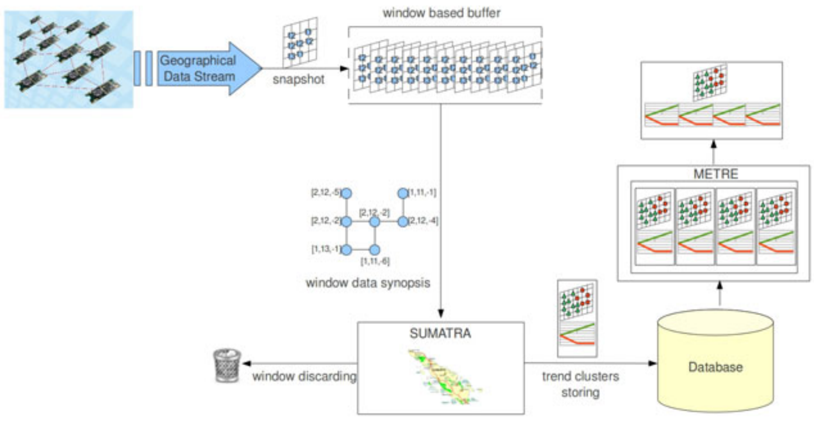


Fig. 3. The stream mining framework TRUST

a clustering technique which models each trajectory as an individual sequence of points generated from a regression model. Unsupervised learning is carried out using an EM technique to cope with the cluster memberships. Nanni and Pedreschi [18] adapted the density-based technique to trajectory data by means of a distance measure between trajectories. Lee et al. [15] proposed a partition-and-group technique to cluster trajectories. This technique partitions a trajectory into a set of segments and groups similar segments into a cluster. However, moving clusters group the moving objects into clusters whose identity remains unchanged, while the cluster location and the content is subject to changes in time. The key difference between a trajectory cluster and a moving cluster is that a trajectory cluster has a constant set of objects throughout its lifetime, while the content of a moving cluster may change over time (i.e. one or more objects may leave the group or new objects may enter it). The seminal technique for discovering moving clusters from a history of recorded trajectories has been proposed in [14]. Spatial clusters are discovered at each snapshot by resorting to a static density-based clustering and the results are then combined in a set of moving clusters. Li et al. [16] proposed the exploitation of the micro-clustering originally proposed by Zhang et al. in [24] and extended it to the moving micro-clustering. They considered a moving micro-cluster as a group of objects that are not only close one to each other at a current time, but are also likely to move together for a while. Finally, research on the identification of areas which remain dense in a period of time [13], given the locations and velocities of the currently moving objects, is related to the discovery moving clusters.

4 Trend Cluster Discovery

TRUST is a stream management framework which permits, both online and offline, the trend cluster discovery in a stream of snapshots. The top-level description of the framework is illustrated in Figure 3. For the online discovery process, a buffer consumes snapshots as they are transmitted from a sensor network and triggers SUMATRA for the intra-window trend cluster discovery. For the offline discovery process, a buffer consumes the window-stamped trend cluster sets stored in the database and pours them window-by-window into METRE. METRE allows to explore offline the trend clusters thus supporting a roll-up navigation of the sensor readings at multiple levels of time granularity. The input parameters of intra-window trend cluster discovery include w that is the number of snapshots which compose a window ($w > 1$) and δ that is the domain similarity threshold. The input parameter of inter-window trend cluster discovery is Ω, which is the number of windows which compose a higher order window ($\Omega > 1$). Both techniques are described in the following sub-sections.

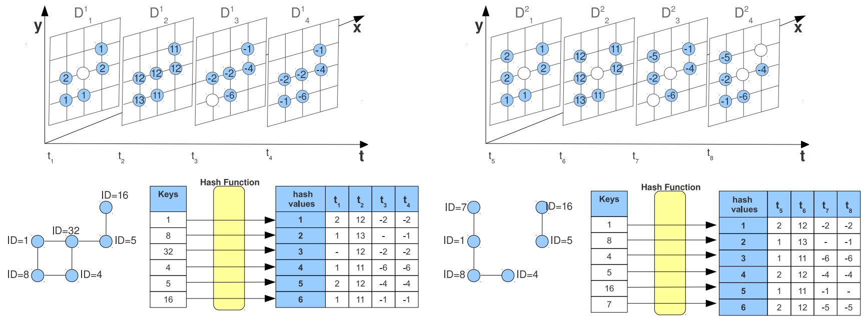
4.1 Online Intra-window Trend Cluster Discovery

SUMATRA processes online a window of snapshots, computes the trend clusters along the time horizon of the window, stores these trend clusters in a database and discards the readings of the processed window. A buffer consumes snapshots as they are periodically transmitted from the sensor network and pours them, window-by-window, into the data synopsis S to be processed by SUMATRA. Details of the the buffer synopsis, the trend cluster structure, the intra-window trend cluster discovery algorithm and the trend cluster storage in a database are described in the following paragraphs.

Buffer Synopsis. Let D be a spatially distributed stream, w be the window size and W_i be the i-th window in D composed of w consecutive snapshots, that is, $W_i = \langle D_1^i, D_2^i, \dots, D_w^i \rangle$. The window W_i is buffered into an instance of the data synopsis S, which comprises the graph structure $G_i(N_i, E_i)$ and the hash table H_i (see Figure 4). The node set N_i maps the set of sensors which transmit a reading in at least one snapshot of W_i. The edge relation E_i models a user-defined spatial closeness relation based on a distance relation (e.g. connected nodes are distant less than a threshold) or a directional relation (e.g., connected nodes are one at the north of the other) between nodes (sensors). The hash table H_i stores the windowed readings in a bi-dimensional table, where each row u maps a sensor (or equivalently a node of N_i) and each column j enumerates a transmission point into the window W_i. Thus, the tabular entry $H_i[u][j]$ stores the reading of the sensor u transmitted at the j-th snapshot (D_j^i) of the window W_i. In the presented framework, the missing readings are on-the-fly replaced in their tabular cells by the median of readings computed on the associated tabular row.

Trend Cluster Definition. The structure of a trend cluster is defined in Definition 1.

Definition 1 (Trend cluster). *Let W_i be the window currently buffered in the data synopsis $S[G_i, H_i]$, the triple $[i, C, P]$ is a trend cluster along the time horizon of W_i iff (i) i is the enumerator of window W_i in the window segmentation of the stream, (ii) C is a subset of N_i which is feasible with respect to E_i (see Definition 2), (iii) P is the trend polyline prototype of C (see Definition 3) and (iv) $[C, P]$ satisfies the property of intra-cluster polyline homogeneity in W_i (see Definition 4).*



Keys
1
8
32
4
5
16

hash values	t_1	t_2	t_3	t_4
1	2	12	-2	-2
2	1	13	-	-1
3	-	12	-2	-2
4	1	11	-6	-6
5	2	12	-4	-4
6	1	11	-1	-1

Keys
1
8
4
5
16
7

hash values	t_5	t_6	t_7	t_8
1	2	12	-2	-2
2	1	13	-	-1
3	1	11	-6	-6
4	2	12	-4	-4
5	1	11	-1	-
6	2	12	-5	-5

Fig. 4. Buffering a window of w (=4) consecutive snapshots into the data synopsis S

The definitions of E-based feasibility, trend polyline prototype and intra-cluster polyline homogeneity are reported as follows.

Definition 2 (E-based cluster feasibility). *Let C be a subset of N_i. C is feasible w.r.t. the edge relation E_i iff $\forall u, v \in C, u$ is E-reachable from v in C (or vice-versa), where u is E-reachable from v iff $\langle u, v\rangle \in E_i$, or $\exists z \in C$ such that $\langle u, z\rangle \in E_i$ and z is E-reachable from v in C.*

Definition 3 (Trend polyline prototype). *Let C be a subset of N_i. The trend polyline prototype of C, denoted by P, is a piecewise straight line passing through the sequence of timestamped vertices represented as follows:*

$$P = \langle (t_1^i, {y_1^i}_C), (t_2^i, {y_2^i}_C), \ldots, (t_w^i, {y_w^i}_C)\rangle, \tag{2}$$

where for each vertex $(t_j^i, {y_j^i}_C)$, t_j^i denotes the timestamp associated to the j-th snapshot D_j^i of the owner window W_i, and ${y_j^i}_C$ denotes the median of the readings at snapshot D_j^i for the nodes grouped in C. In the practice, ${y_j^i}_C = median(\{H_i[u][j] \mid u \in C\})$, with $H_i[u][j]$ the corresponding tabular entry of H_i.

Definition 4 (Intra-cluster polyline homogeneity). *Let C be a subset of N_i, and P be the polyline prototype of C. The intra-cluster polyline homogeneity of $[C, P]$ is a Boolean property defined as follows:*

$$homogeneity([C,P]) = \begin{cases} 1 \text{ iff} \displaystyle\sum_{u \in C} sim(u,P) = \sharp C \\ 0 \qquad\quad otherwise \end{cases} \tag{3}$$

where $sim(u,P) = \begin{cases} 1 & \text{iff} \forall j = 1, \ldots, w : |(H[u][j] - {y_j^i}_C)| \le \delta \\ 0 & otherwise \end{cases}$, $\sharp C$ is the cardinality of C and δ represents the user-defined domain similarity threshold.

Algorithm 1. SUMATRA$(i, S[G_i, H_i], \delta, DB)$

```
– main routine
Require: $i$: the number which enumerates the window $W_i$ in the stream
Require: $S[G_i, H_i]$: the data synopsis where $W_i$ is buffered
Require: $\delta$: the user-defined domain similarity threshold
Require: $DB$: the database where trend clusters computed in $W_i$ are stored
 1: $h \leftarrow 1$
 2: for all $u \in N_i$ do
 3:   if $u$ is $UNCLUSTERED$ then
 4:     $[C_h, P_h] \leftarrow$ expandTrendCluster($\{u\}$,polylinePototype($\{u\}$), $u$)
 5:     store($DB$, $[i, C_h, P_h]$)
 6:     $h \leftarrow h + 1$
 7:   end if
 8: end for
– expandTrendCluster $(C_h, P_h, u) \mapsto [C_h, P_h]$
 1: $\eta(u) \leftarrow$ neighborhood($u$)
 2: $[tempC, tempP] \leftarrow [C_h \cup \eta(u)$, polylinePrototype$(C_h \cup \eta(u))]$
 3: if homogeneity$(tempC, tempP)$ then
 4:   $[C_h, P_h] \leftarrow [tempC, tempP]$
 5:   for all $v \in \eta(u)$ do
 6:     $[C_h, P_h] \leftarrow$ expandTrendCluster$(C_h, P_h, v)$
 7:   end for
 8: else
 9:   for all $v \in \eta(u)$ do
10:     $[tempC, tempP] \leftarrow [C_h \cup q$, polylinePrototype$(C_h \cup v)]$
11:     if polylinePurity$(tempC, tempP)$ then
12:       $[C_h, P_h] \leftarrow$ expandTrendCluster$(tempC, tempP, v)$
13:     end if
14:   end for
15: end if
```

SUMATRA. Trend clusters are discovered by partitioning nodes of N_i into subsets (clusters) which are completely connected by the edges of E_i. The top-level description of SUMATRA is reported in Algorithm 1. An unclustered node u is randomly chosen as a seed for the construction of a new trend cluster $[C_h, P_h]$ and is added to the initially empty cluster C_h (lines 2-3 in the $main\ routine$). Then $[C_h, P_h]$ is expanded (line 4 in the $main\ routine$) as long as the output trend cluster satisfies the properties of both E-based cluster feasibility (see Definition 2) and intra-cluster polyline homogeneity (see Definition 4). The completely constructed trend cluster $[i, C_h, P_h]$ is then stored in the database DB (line 5 in the $main\ routine$). The trend cluster expansion process is performed by calling the subroutine $expandTrendCluster()$ in Algorithm 1. The expansion process for $[C_h, P_h]$ is driven by a seed node u and it is recursively defined. First, the strong neighborhood $\eta(u)$ is constructed (line 1 in the $expandTrendCluster()$). This neighborhood is computed by considering the unclustered nodes of N_i, which are directly reachable from u by means of the edges of E_i which are labeled as strong. An edge of E_i is strong iff it connects nodes whose readings differ at worst δ one-to-each-

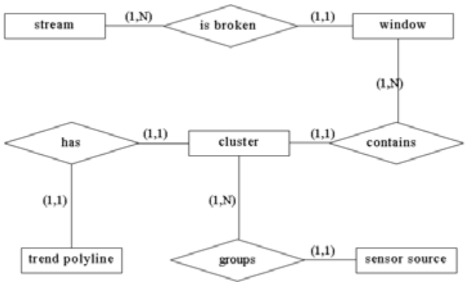


Fig. 5. Entity-relationship schema of database where intra-window trend clusters are stored

other on the consecutive snapshots of the window. Once $\eta(u)$ is computed, the candidate cluster $tempC = C_h \cup \eta(u)$ and the associated trend polyline prototype $tempP$ are computed (line 2 in the $expandTrendCluster()$). The polyline prototype is computed as reported in Definition 3. The intra-cluster polyline homogeneity of $[tempC, tempP]$ is computed (line 3 of $expandCluster()$). Then two cases are distinguished.

1. The candidate $[tempC, tempP]$ satisfies the intra-cluster trend homogeneity property, then nodes of $\eta(u)$ are clustered into C_h and the last computed $tempP$ is assigned to T_h (lines 4-7 in the $expandTrendCluster()$).
2. The candidate $[tempC, tempP]$ does not satisfy the intra-cluster trend homogeneity property and the addition of each node of $\eta(u)$ to C_h is evaluated node-by-node (lines 9-14 in the $expandTrendCluster()$).

In both cases, nodes newly clustered in C_h are iteratively chosen as seeds to continue the expansion process (line 6 and line 12 in the $expandTrendCluster()$). The expansion process stops when no new node is added to the cluster.

Storage of trend clusters in database. Let $[i, C_h, P_h]$ be a trend cluster discovered in W_i, SUMATRA stores the window identifier i, the identifier of the nodes which are grouped in C_h and the series of vertices of the trend polyline prototype P_h in the database DB. The window identifier is the progressive enumerative of the window. The trend polyline is stored in DB as the array of w median values composing P_h. We can avoid storing the time coordinates as they are equally spaced in time. The conceptual schema according to which trend clusters are arranged in DB is reported in Figure 5.

4.2 Offline Inter-window Trend Cluster Discovery

METRE is triggered by the user(s) request. It retrieve the sets of trend clusters stored in DB for consecutive windows and computes new trend clusters from these sets such that the output trend clusters arise along larger windows. These windows (higher order windows) are obtained by sequencing the time horizon of Ω consecutive windows pasty processed by SUMATRA. The discovery process is offline with respect to the stream, as past readings were discarded after passing through SUMATRA. Due to the offline nature of the discovery process, METRE can be run again and again by varying Ω.

Let $TC[i]^w = \{[i, C_h, P_h]\}_h$ be the set of trend clusters $[i, C_h, P_h]$ stored in DB for the window W_i. A buffer iteratively consumes the sets of trend clusters retrieved in DB for the Ω consecutive windows and triggers METRE for the inter-window trend cluster discovery. At the j-th iteration, the series of trend cluster sets composed as follows:

$$TC[(j-1)\Omega+1]^w, TC[(j-1)\Omega+2]^w, \ldots, TC[(j-1)\Omega+(\Omega-1)]^w, TC[j\Omega]^w$$

is poured into Algorithm 2. METRE processes this series $\langle TC[(j-1)\Omega+k]^w \rangle_{k=1,2,\ldots\Omega}$ and outputs a new set of trend clusters, denoted as $\widetilde{TC}[j]^{w\Omega}$, which contains trend clusters along the time horizon of the window of D denoted as $\widetilde{W}_j$. As the time horizon of $\widetilde{W}_j$ is that of the j-th window of the $w \times \Omega$ window segmentation of D, $\widetilde{W}_j$ covers time horizons of windows in $\langle W_{(j-1)\Omega+k} \rangle_{k=1,2,\ldots\Omega}$ (see line 2 of the $main\ routine$).

In METRE, the computation of the output $\widetilde{TC}[j]^{w\Omega}$ is inspired by the consideration that sensors, which are repeatedly grouped together in a cluster for consecutive windows $W_{(j-1)\Omega+k}$ (with k ranging between 1 and Ω), transmit readings which evolve with a similar trend prototype along the window $\widetilde{W}_j$.

Let $\widetilde{N}$ be the set of sensors which appear in at least one cluster of the input series $\langle TC[(j-1)\Omega+k]^w \rangle_{k=1,2,\ldots\Omega}$ (line 3 of the $main\ routine$). The construction of each new trend cluster $[j, \widetilde{C}_h, \widetilde{P}_h]$ along the time horizon of $\widetilde{W}_j$ starts by choosing a seed node $u \in \widetilde{N}$ which is not yet clustered along $\widetilde{W}_j$ (line 5 of the $main\ routine$). Let $\widetilde{C}_k^{[u]}$ be the cluster of the set $TC[(j-1)\Omega+k]^w$ which contains u (line 7 of the $main\ routine$) and $\widetilde{P}_k^{[u]}$ be the trend polyline prototype associated to $\widetilde{C}_k^{[u]}$ in the set $TC[(j-1)\Omega+k]^w$ (line 8 of the $main\ routine$). The cluster $\widetilde{C}_h$ initially contains u (line 10 of the $main\ routine$). Then it is expanded by grouping the nodes of $\widetilde{N}$ which are spatially close to u and are clustered in the same way as u along the series of windows $W_{(j-1)\Omega+k}$ sequenced into $\widetilde{W}_j$ (line 11 of the $main\ routine$). Once $\widetilde{C}_h$ is completely expanded, $\widetilde{P}_h$ is built by sequencing the trend polyline prototypes $\widetilde{P}_k^{[u]}$ with k ranging between 1 and Ω (line 12 of the $main\ routine$). Finally, the computed trend cluster $[j, \widetilde{C}_h, \widetilde{P}_h]$ is added to the set $\widetilde{TC}_j^{w\Omega}$ (line 13 of the $main\ routine$).

The cluster expansion (see $expandCluster()$ into Algorithm 2) is driven by an expansion seed node u. The cluster C_h is expanded by adding any unclustered node $v \in \widetilde{N}$ which is spatially close to the current seed u and which is grouped into $\widetilde{C}_k^{[u]}$ for each k ranging between 1 and Ω (lines 1-5 of $expandCluster()$). The cluster expansion process is recursively called by considering each new node grouped in $\widetilde{C}_h$ as an expansion seed (line 3 of $expandCluster()$).

5 Experiments

TRUST is written in Java and interfaces a MySQL DBMS. In this Section, we present results of the empirical evaluation of the correctness and relevance to the human interpretation of the trend clusters discovered by SUMATRA and METRE within TRUST. The trend cluster visualizer TREC-Vis, now integrated in TRUST, is used to plot the trend clusters. The goal of the experiments is two-fold.

Algorithm 2. METRE: $\langle TC[(j-1)\Omega+k]^w\rangle_{k=1,2,\ldots\Omega} \mapsto \widetilde{TC}[j]^{w\Omega}$

```
– main routine
Require: $TC[(j-1)\Omega+k]^w$: the set of trend clusters along the $((j-1)\Omega+k))$-th window of the $w$-sized segmentation of $D$ with $k$ ranging between 1 and $\Omega$
Ensure: $\widetilde{TC}[j]^{w\Omega}$: the set of trend clusters along the $j$-th window of the $w \times \Omega$ segmentation of $D$
 1: $\widetilde{TC}[j]^{w\Omega} \leftarrow \oslash$
 2: $\widetilde{W}_j \leftarrow$mergeWindows($\{W_{(j-1)\Omega+k}\}_k$)
 3: $\widetilde{N} \leftarrow \bigcup_{k=1}^{\Omega} \left( \bigcup_{[(j-1)\Omega+k,c,p] \in TC[(j-1)\Omega+k]^w} c \right)$
 4: $h \leftarrow 1$;
 5: for all ($u \in \widetilde{N}$ and $u$ is unclustered) do
 6:     for all ($k = 1$ TO $\Omega$) do
 7:         $\widetilde{C}_k^{[u]} \leftarrow$determineCluster($u, TC[(j-1)\Omega+k]$)
 8:         $\widetilde{P}_k^{[u]} \leftarrow$determineTrendPolyline($\widetilde{C}_k^{[u]}, TC[(j-1)\Omega+k]$)
 9:     end for
10:     $\widetilde{C}_h = \{u\}$;
11:     $\widetilde{C}_h \leftarrow$ expandCluster($u, \widetilde{C}_h, \widetilde{N}, \langle\widetilde{C}_k^{[u]}\rangle_k$)
12:     $\widetilde{P}_h = \widetilde{P}_1^{[u]} \bullet \widetilde{P}_2^{[u]} \bullet \ldots \bullet \widetilde{P}_\Omega^{[u]}$;
13:     $\widetilde{TC}[j]^{w\Omega} = \widetilde{TC}[j]^{w\Omega} \cup [j, \widetilde{C}_h, \widetilde{P}_h]$;
14: end for
– expandCluster($u, \widetilde{C}_h, \widetilde{N}, \langle\widetilde{C}_k^{[u]}\rangle_k) \mapsto \widetilde{C}_h$
 1: for all ($v \in \widetilde{N}$ with $v$ spatially close to $u$ and $v$ unclustered) do
 2:     if belongCluster($v$, $\langle\widetilde{C}_k^{[u]}\rangle_k$) then
 3:         $\widetilde{C}_h \leftarrow$ expandCluster( $v, \widetilde{C}_h \cup \{v\}, \langle\widetilde{C}_k^{[u]}\rangle_k$) );
 4:     end if
 5: end for
```

1. We evaluate correctness and human interest towards the trend clusters discovered by SUMATRA. We first consider an artificially generated stream and check the ability of SUMATRA of detecting underlying known trend cluster configurations. Then, we evaluate the trend clusters which are discovered for sumarization scope in real data streams. In this case, we study learning time, accuracy of summarization and size of trend clusters by varying the window size w and the similarity threshold δ. As baseline for the evaluation of SUMATRA results, we consider the density based clustering performed snapshot by snapshot (with $w = 1$). Finally, we show the relevance of discovered trend clusters to a human interpretation by showing how the visualization of trend clusters may reveal knowledge such as data trends, stability or drift of the cluster shape in time and permits a readable visualization of the stream when few cluster polylines are plotted in place of several sensor polylines.
2. We empirically show that trend cluster discovery performed by METRE is able to capitalize on trend clusters, window-by-window stored in databases, by resulting more efficient in learning time than SUMATRA which processes original readings from the stream. We observe that the improved efficiency of the learning

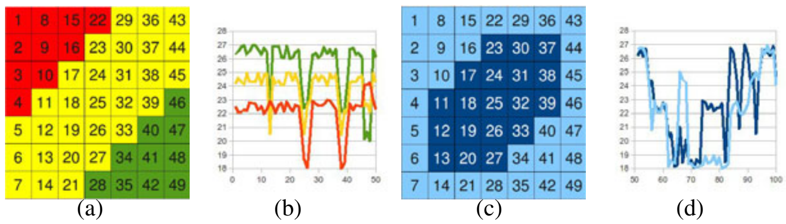


Fig. 6. The trend-cluster configurations according to $Artificial 49 \times 49$ is generated

process causes only a low decrease of summarization accuracy. Additionally, we observe that METRE permits to compute offline trend clusters along any horizon time (which is multiple of the window time), although the readings are now discarded.

In the following sub-section, we describe the streams employed in this study and the experimental setting. Subsequently, we comment results obtained with these streams.

5.1 Streams and Experimental Setting

Experiments are run on Intel(R) Core(TM) 2 DUO CPU $E4500@2.20$GHz with 2.0 GiB of RAM Memory, Ubuntu 10.04 (lucid) Kernel Linux $2.6.32 - 26 - generic$.

Data Streams. $Artificial 49 \times 49$ collects data artificially generated for a network of 49 virtual sensors distributed over a squared 7×7 grid (see Figure 6). A sensor is considered to be close to the neighbor sensors which are located in the grid cells around the sensor (up, down, left and right). Readings are generated with range in $[18, 27]$. The stream is obtained by sequencing 50 snapshots which underly the trend cluster configuration in Figures 6(a)-6(b) and 50 snapshots which underly the trend cluster configuration in Figures 6(c)-6(d). Readings of each sensor vary according to the trend polyline of the associated cluster with a random error uniformly distributed between 0 and 1. The *Intel Berkeley Lab*[1] data stream collects temperature (in degrees Celsius) and humidity (in RH) readings transmitted every 31 seconds from 54 sensors deployed in the Intel Berkeley Research lab between February 28th and April 5th 2004. The stream includes about 65000 snapshots. A sensor is considered spatially close to every other sensor within a range of six meters. Readings are affected by noise and outliers. Missing values occur in most snapshots. Thus, the number of sensors is truly variable in time. By using a box plot, we observe that the air temperature values presumably range between 9.75 and 34.6, while the humidity values presumably range between 0 and 100. The *South American Climate*[2] collects monthly-mean air temperature measurements (in degrees Celsius) recorded between 1960 and 1990 and interpolated over a 0.5 degree by 0.5 degree latitude/longitude grid of South America. The grid nodes are centered on 0.25 degree for a total of 6477 sensors. The number of nearby stations that affect a

[1] `http://db.csail.mit.edu/labdata/labdata.html`
[2] `http://climate.geog.udel.edu/~climate/html_pages/archive.html`

grid-node estimate is twenty on average. This resulted in more realistic air-temperature fields. The stream includes 360 snapshots. No missing values are in the stream. A sensor is considered spatially close to the sensors which are located in the surrounding cells of the grid. The air temperature values range between -7.6 and 32.9.

Evaluation Measures. Let D be a stream, w be a window size and $TC = \langle\{i, C_h, P_h\}_h\rangle_i$ be the series of trend cluster sets discovered window-by-window in the w-sized window of D. In particular, the set $\{i, C_h, P_h\}_h$ denotes the set of trend clusters discovered in the i-th w-sized window W_i of D and stored in the database DB. The accuracy of TC in compactly representing D is evaluated by means of the mean absolute error (mae) which is performed when the stream D is approximately reconstructed (as $\hat{D}$) from TC. The mean absolute error measures how forecasts obtained by trend clusters are close to the real outcome. The use of the mae is motivated by the fact that this error measure is the least sensitive to the presence of outliers. Formally,

$$mae(D, \hat{D}) = \frac{\sum_{u,t} |v[u][t] - \hat{v}[u][t]|}{\sharp D} \tag{4}$$

where $\hat{D}$ is the stream reconstructed from TC. For each sensor u and for each transmission time point t, the predicted value $\hat{v}[u][t]$ is the median coordinate of the vertex timestamped with t in the trend polyline prototype P, associated to the cluster C. C is the cluster which groups u along the horizon time of W_i and W_i is the window within the time point t falls . Formally, let $[i, C, P]$ be a trend cluster stored in DB for the window W_i, such that $t \in W_i$ and $u \in C$. Then $\hat{v}[u][t] = P[t].MedianValue$. It is noteworthy that the lower the mae, the more accurate the TC.

The summary rate ($\sigma^{\%}$) is the percentage computed as the ratio of the size (in bytes) of TC to the size (in bytes) of D, that is:

$$\sigma^{\%}(D \rightarrow TC) = \frac{size(TC)}{size(D)} \times 100\% \tag{5}$$

where $size(\cdot)$ is the memory occupation in bytes. The lower $\sigma^{\%}$, the more compact TC in summarizing D.

Finally, the computation time is the time (in milliseconds) spent on average in computing any (intra-window or inter-window) trend cluster set.

5.2 Intra-window Trend Cluster Discovery Evaluation

To evaluate the correctness of the discovered trend clusters we run SUMATRA on the artificial data stream $Artificial49 \times 49$ and compare the detected trend clusters with the expected ones. Experiments are run with $w = 20$, $\delta = 2.0$ for a total of five windows. Clusters and trends discovered, window-by-window, by SUMATRA are plotted in Figure 7. These trend clusters are plotted by using TREC-Vis. As expected, SUMATRA correctly detects the expected trend cluster configuration reported in Figure 6(a) in the windows W_1 and W_2 and the expected trend cluster configuration reported in Figure 6(c) in the windows W_4 and W_5. The trend cluster configuration detected over

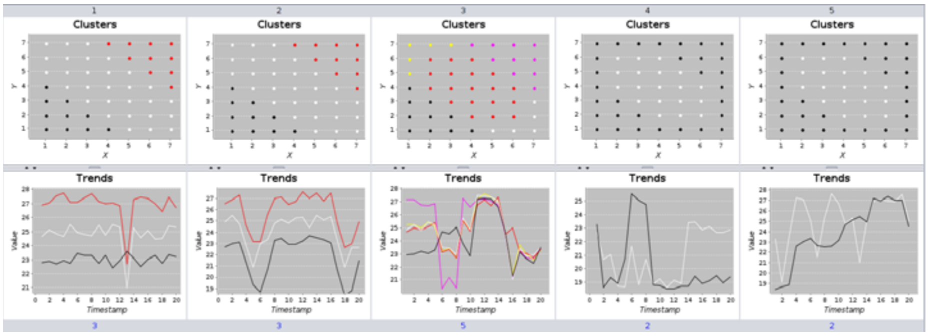


Fig. 7. Trend clusters discovered by SUMATRA with w=20 and $\delta = 2.0$

Table 1. *Artificial49×49*: mae with $w = 20$ and $\delta = 2.0$

W_1	W_2	W_3	W_4	W_5
0.2355108066	0.2402879404	0.2268378102	0.2393314381	0.239197659

the central window W_3 reveals the drift occurring between these two configurations (W_3 includes 10 snapshots generated according to the configuration in Figure 6(a) followed by 10 snapshots generated according to the configuration in Figure 6(c)). To adapt the output to the drift of cluster shape, SUMATRA transfers the sensors identified by 1, 2, 3, 8 and 15 from the central cluster to a new cluster (yellow cluster in W_3) and the sensors identified by 35, 42, 47, 48 and 49 from the central cluster to a separate cluster (white cluster in W_3). This way, SUMATRA still identifies groups of spatially close sensors whose readings vary with a trend that is approximately the same over the window under consideration. It is noteworthy that although these two clusters are associated to overlapped trend polyline, they are distinct clusters due to the spatial discontinuity of the grouped sensors. A further consideration concerns the fact that the plot of trend clusters provides an interpretable insight of the space-time distribution of readings. Table 1 collects the mean absolute error (mae) of trend-cluster sets discovered in $Artificial49 \times 49$. The mae is alway less than 0.24 for each window, thus confirming that the trend clusters are accurate if employed as summarization patterns.

Further experiments are performed by running SUMATRA with real data streams by varying the window size w and the domain similarity threshold δ as reported in Table 5.2. δ ranges among 5%, 10% and 20% of the expected domain range of the streamed dimension. Experiments are run with $w = 1$ to obtain traditional spatial clusters (without trends) as baseline of SUMATRA results. It is a baseline against which the results can be compared. The analysis of the mean absolute error, the summary rate and the average computation time, plotted in Figure 8 for each stream in this study, permits us to make several considerations. In particular, the mae is always significantly below the domain similarity threshold δ (i.e. trend clusters are accurate in compactly representing the stream). The summary rate is always below 100% (i.e. trend clusters summarize the stream). The average window learning time is always less than 1.25 seconds in the case

Table 2. The parameter setting to run SUMATRA. The trend clusters computed in the bold-valued setting are then used to evaluate METRE.

Stream	Dimension	δ	w
Intel Berkeley Lab	Temperature	1.25, **2.5**, 5.0	1, 256, 512, 1024
Intel Berkeley Lab	Humidity	5, **10**, 20	1, **256**, 512, 1024
South American Climate	Temperature	2, **4**, 8	1, **6**, 12, 24

of the *Intel Berkeley Lab* stream (both for Temperature and Humidity), and it is always less than 93 seconds in the *South American Climate* stream. In general, trend clusters discovered window by window ($w > 1$) summarize a stream better than spatial clusters discovered snapshot by snapshot (w = 1). Indeed, the accuracy of the trend cluster summarization is better than the accuracy of the spatial cluster summarization. Final considerations are on the window average computation time. As the computation time depends on the window size, computing trend clusters in a window is more expensive in time than computing spatial clusters in a snapshot. On the other hand, the total time spent to compute spatial clusters on the consecutive snapshots of a window is more than the time spent to compute the trend clusters on the entire window. The observed low computation time justifies our consideration of SUMATRA as a system that processes sensor readings in real-time, that is, when a new window is completed in the stream, the trend-cluster discovery in the past window is already completed.

Different considerations are suggested by analyzing trend clusters obtained by varying w and/or δ. By increasing w, the mean absolute error decreases (i.e. trend clusters are more accurate), while the summary rate does not necessarily decrease. At the same time, we observe that the average computation time increases with the window size as more data are processed at the same time, but the analysis of the computation time spent totally in processing the entire stream is not significantly affected by the window size. By increasing δ, the mean absolute error of the trend clusters increases slightly but, as expected, the summary rate of the stream is always lower. The tuning of δ is clearly in charge of determining the best trade-off between the accuracy and the compression rate. Moreover, we observe that the average computation time increases with δ as the size of the neighborhoods explored to construct the clusters increases with δ.

Finally we observe that, experiments performed with the *Intel Berkeley Lab* stream confirm that SUMATRA is able to deal with noisy data, outliers and networks exhibiting a number of sensors which may change in time. Each noisy sensor is modeled as a singleton cluster (the red cluster in W_{11} and W_{12} of Figure 9) which is associated to a wiggly trend polyline. The trend reveals the time points where the abrupt change of trend occurs. Experiments performed with the *South American Air Climate* stream confirm that SUMATRA is scalable on large networks.

5.3 Inter-window Trend Cluster Discovery Evaluation

We assume that trend clusters computed intra-window by SUMATRA with the parameter setting reported in boldface in Table 5.2 are stored in a MySQL database and processed offline to generate new trend clusters along higher order windows. We

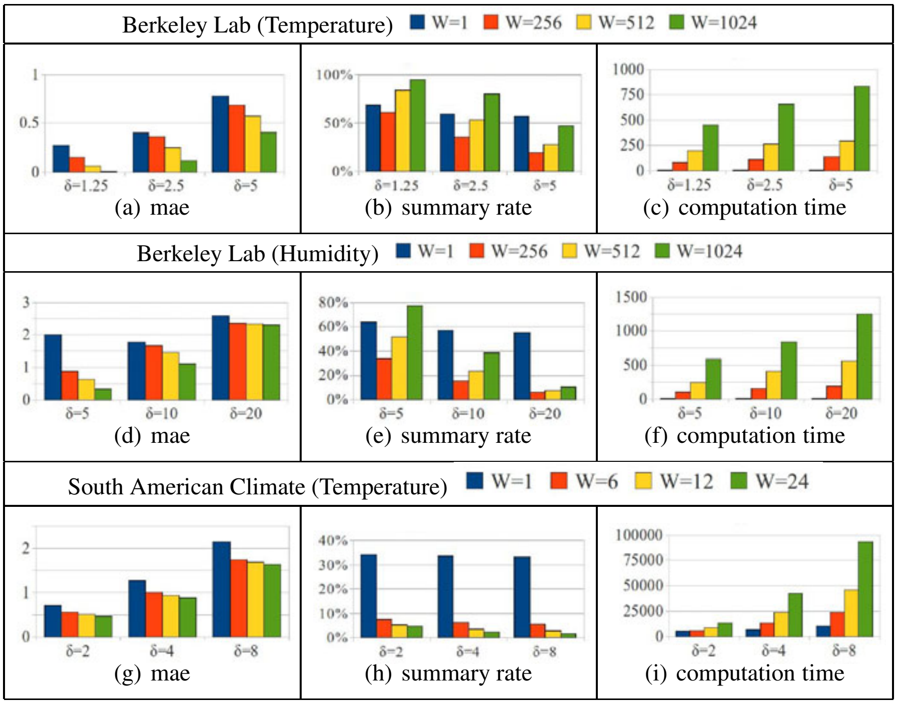


Fig. 8. Online intra-window trend cluster discovery: mean absolute error, summary rate and average computation time are plotted (Y axis) by varying window size (X axis). SUMATRA is run by varying similarity domain threshold δ and window size w.

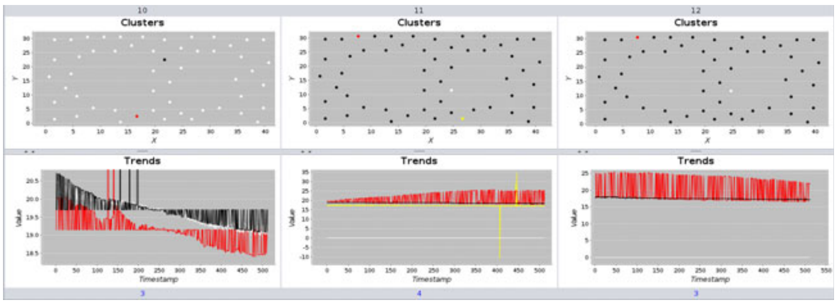


Fig. 9. Intel Berkeley Lab (Temperature): visualization of trend clusters discovered over three consecutive windows with $w = 256$ and $\delta = 2.5$. This plot reveal the presence of a big cluster which collects almost all sensors and noisy sensors at each window.

run METRE with $\Omega = 4$ and compare the mean absolute error, the summary rate and the average learning time of METRE with that of SUMATRA run on the $w \times \Omega$-sized window segmentation of the same stream. As METRE approximates the trend cluster discovery process performed by SUMATRA, we intend to evaluate how good the

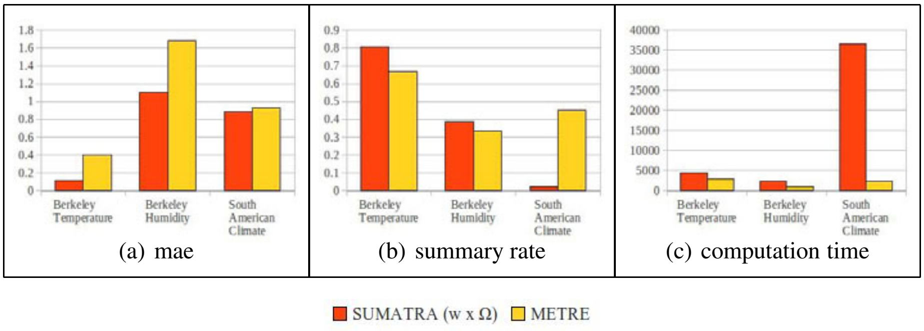


Fig. 10. Offline inter-window trend cluster discovery: mean absolute error, summary rate and average computation time are plotted (Y axis) by varying the stream (X axis). METRE is run with $\Omega = 4$. METRE is compared with SUMATRA run on the $w \times \Omega$ window segmentation of the same stream.

Table 3. The total number of trend clusters discovered on the entire stream. METRE is run with $\Omega = 4$. SUMATRA is run with window size=$w \times \Omega$.

Stream	Berkeley Lab		South American Climate
Stream Attribute	Temperature	Humidity	Temperature
Experimental Setting	$w = 256$ $\delta = 10$	$w = 256$ $\delta = 2.5$	$w = 6$ $\delta = 4$
METRE	1875	917	7580
SUMATRA	2199	1058	1231

approximation process is. The results plotted in Figure 10 show that METRE outputs trend clusters with a slight increase of the mean absolute error with respect to that of trend clusters output by SUMATRA (i.e. the accuracy of trend clusters decreases). In any case, the mean absolute error remains under δ. On the other hand, as expected, the average computation time spent for the discovery process significantly decreases from SUMATRA to METRE. This confirms our intuition that processing trend clusters is a less complex task than directly processing sensor readings. However, the analysis of the summary rate is not subjected to a unique interpretation. In fact, the compression rate does not significantly change from SUMATRA to METRE when considering the Intel Berkeley Lab stream. This is confirmed by the total number of trend clusters discovered on the entire stream which is almost the same with both SUMATRA and METRE (see Table 5.3). At the same time, the number of clusters discovered by METRE is greater than the corresponding number of trend clusters discovered by SUMATRA in South American Climate stream. In this case, several clusters formed by a single node are found. In any case, trend clusters remain more compact than the stream (the summary rate is always significantly less than 100%). So we can conclude that trend clusters discovered by METRE closely approximate trend clusters which would be originally discovered by SUMATRA. Furthermore, trend clusters are discovered offline and they can be discovered with several time resolution by running METRE with a different values of Ω.

6 Conclusions

The paper presents a stream management system called TRUST, which supports online and offline trend cluster discovery from spatially distributed readings of a sensor network. A stream is segmented into consecutive, equally sized windows. Once a window is completely flowed in the stream it is poured into the online learning component, called SUMATRA, which partitions the window into a set of trend clusters and store discovered trend clusters in a database as a compact and accurate representation (summary) of the window. Sensor readings passed trough SUMATRA are discarded. Subsequently, each window-stamped set of trend clusters can be queried in the database and triggers METRE for the inter-window trend cluster discovery. METRE operates offline and computes trend clusters by merging clusters and trends previously computed along consecutive windows and stored in the database. An empirical study of both SUMATRA and METRE is illustrated. The conclusions drawn from this study are twofold. The former is that the intra-window, online trend clusters, discovered by SUMATRA are an accurate, compact representation of the stream which is truly computed in real time. This consideration is not affected by the learning parameter tuning (window size and/or domain similarity). The latter is that inter-window, offline trend clusters discovered by METRE can be computed in a low computation time by processing trend cluster sets already stored in the database. The offline process can be repeated by varying the number of windows to be merged. The output trend clusters, once again, are a compact, accurate representation of the stream which can eventually be stored in the database for future analysis. As future work we plan to investigate how combine/discover trend clusters in scenarios where sensors measure several dimensions at once.

Acknowledgments. This work is supported by the project "Multi-relational approach to spatial data mining" funded by the University of Bari ”Aldo Moro” and the project "EMP3: Efficiency Monitoring of Photovoltaic Power Plants" funded by "Fondazione Cassa Di Risparmio Di Puglia". The authors gratefully acknowledge Lynn Rudd for reading the final version of this paper.

References

1. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for clustering evolving data streams. In: Aberer, K., Koubarakis, M., Kalogeraki, V. (eds.) VLDB 2003. LNCS, vol. 2944, pp. 81–92. Springer, Heidelberg (2004)
2. Aggarwal, C.C., Yu, P.S.: A survey of synopsis construction in data streams. In: Data Streams: Models and Algorithms, vol. 31, pp. 170–207. Springer, Heidelberg (2007)
3. Armenakis, C.: Estimation and organization of spatio-temporal data. In: Frank, A.U., Formentini, U., Campari, I. (eds.) GIS 1992. LNCS, vol. 639. Springer, Heidelberg (1992)
4. Babcock, B., Datar, M., Motwani, R., O'Callaghan, L.: Maintaining variance and k-medians over data stream windows. In: PODS 2003, pp. 234–243. ACM, New York (2003)
5. Bhaduri, K., Sivakumar, K.D.K., Kargupta, H., Wolff, R., Chen, R.: In: Aggarwal, C.C. (ed.) Data Streams. Advances in Database Systems, vol. 31, pp. 309–332. Springer, US (2007)
6. Cao, F., Ester, M., Qian, W., Zhou, A.: Density-based clustering over an evolving data stream with noise. In: Ghosh, J., Lambert, D., Skillicorn, D.B., Srivastava, J. (eds.) SIAM SDM 2006 (2006)

7. Chang, W., Zeng, D., Chen, H.: A stack-based prospective spatio-temporal data analysis approach. Decis. Support Syst. 45(4), 697–713 (2008)
8. Chen, Y., Tu, L.: Density-based clustering for real-time stream data. In: KDD 2007, pp. 133–142. ACM, New York (2007)
9. Ciampi, A., Appice, A., Malerba, D.: Summarization for geographically distributed data streams. In: Setchi, R., Jordanov, I., Howlett, R.J., Jain, L.C. (eds.) KES 2010. LNCS, vol. 6278, pp. 339–348. Springer, Heidelberg (2010)
10. Gaber, M.M., Zaslavsky, A., Krishnaswamy, S.: Mining data streams: a review. ACM SIGMOD Record 34(2), 18–26 (2005)
11. Gaffney, S., Smyth, P.: Trajectory clustering with mixtures of regression models. In: KDD 1999, pp. 63–72. ACM, New York (1999)
12. Guha, S., Mishra, N., Motwani, R., O'Callaghan, L.: Clustering data streams. In: FOCS, pp. 359–366 (2000)
13. Hadjieleftheriou, M., Kollios, G., Gunopulos, D., Tsotras, V.J.: On-line discovery of dense areas in spatio-temporal databases. In: Hadzilacos, T., Manolopoulos, Y., Roddick, J., Theodoridis, Y. (eds.) SSTD 2003. LNCS, vol. 2750, pp. 306–324. Springer, Heidelberg (2003)
14. Kalnis, P., Mamoulis, N., Bakiras, S.: On discovering moving clusters in spatio-temporal data. In: Anshelevich, E., Egenhofer, M.J., Hwang, J. (eds.) SSTD 2005. LNCS, vol. 3633, pp. 364–381. Springer, Heidelberg (2005)
15. Lee, J.-G., Han, J., Whang, K.-Y.: Trajectory clustering: a partition-and-group framework. In: SIGMOD 2007, pp. 593–604. ACM, New York (2007)
16. Li, Y., Han, J., Yang, J.: Clustering moving objects. In: KDD 2004, pp. 617–622. ACM, New York (2004)
17. Munro, R., Chawla, S.: An integrated approach to mining data streams. In: Technical Report, University of Sydney. School of Information Technologies (2004)
18. Nanni, M., Pedreschi, D.: Time-focused clustering of trajectories of moving objects. J. Intell. Inf. Syst. 27(3), 267–289 (2006)
19. O'Callaghan, L., Meyerson, A., Motwani, R., Mishra, N., Guha, S.: Streaming-data algorithms for high-quality clustering. In: ICDE, p. 685. IEEE, Los Alamitos (2002)
20. Shekhar, S., Chawla, S.: Spatial databases: A tour. Prentice Hall, Englewood Cliffs (2003)
21. Tobler, W.: Cellular geography. Philosophy in Geography, 379–386 (1979)
22. Vlachos, M., Gunopoulos, D., Kollios, G.: Discovering similar multidimensional trajectories. In: ICDE 2002, p. 673. IEEE Computer Society, Los Alamitos (2002)
23. Wan, L., Ng, W., Dang, X., Yu, P., Zhang, K.: Density-based clustering of data streams at multiple resolutions. Trans. Knowl. Discov. Data 3(3), 1–28 (2009)
24. Zhang, T., Ramakrishnan, R., Livny, M.: Birch: an efficient data clustering method for very large databases. SIGMOD Rec. 25(2), 103–114 (1996)

Author Index

Appice, Annalisa 142
Atzmueller, Martin 79

Benz, Dominik 79

Cayci, Aysegul 119
Ciampi, Anna 142

Eibe, Santiago 119

Fricke, Peter 99

Hilts, Andrew 20
Hotho, Andreas 79

Jungermann, Felix 99

Kisilevich, Slava 57

Last, Mark 57

Malerba, Donato 142
Manhardt, Alexander 1
Menasalvas, Ernestina 119
Mitzlaff, Folke 79
Morik, Katharina 99

Nygren, Else 37

Piatkowski, Nico 99
Prinz, Vivian 1

Saygin, Yucel 119
Schulze, Florian 1
Spinczyk, Olaf 99
Stolpe, Marco 99
Streicher, Jochen 99
Stumme, Gerd 79

Woerndl, Wolfgang 1

Yu, Eric 20