Martin Atzmueller
Andreas Hotho (Eds.)

Mining Ubiquitous and Social Environments (MUSE 2010)

International Workshop at
the European Conference on Machine Learning and
Principles and Practice of Knowledge Discovery in Databases
in Barcelona, Spain, September 20th, 2010

Table of Contents

Preface

The emergence of ubiquitous computing has started to create new environments consisting of small, heterogeneous, and distributed devices that foster the social interaction of users in several dimensions. Similarly, the upcoming social semantic web also integrates the user interactions in social networking environments. Mining in ubiquitous and social environments is thus an emerging area of research focusing on advanced systems for data mining in such distributed and network-organized systems. It also integrates some related technologies such as activity recognition, Web 2.0 mining, privacy issues and privacy-preserving mining, predicting user behavior, etc.

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. However, the characteristics of ubiquitous and social mining are in general quite different from the current mainstream data mining and machine learning. Unlike in traditional data mining scenarios, data does 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. Steps into this new and exciting application area are the analysis of this new data, the adaptation of well known data mining and machine learning algorithms and finally the development of new algorithms.

The goal of this workshop is to promote an interdisciplinary forum for researchers working in the fields of ubiquitous computing, social semantic web, Web 2.0, and social networks which are interested in utilizing data mining in an ubiquitous setting. The workshop seeks for contributions applying state-of-the-art mining algorithms on ubiquitous and social data. Papers focusing on the intersection of the two fields are especially welcome. In short, we want to accelerate the process of identifying the power of advanced data mining operating on data collected in ubiquitous and social environments, as well as the process of advancing data mining through lessons learned in analyzing these new data.

This proceedings volume comprises the papers of the MUSE 2010 workshop. In total, we received 18 submissions, from which we were able to accept seven submissions based on a rigorous reviewing process. Additionally, the workshop features an invited talk on the convergence of social and ubiquitous data.

Based on the set of accepted papers, and the invited talk, we set up four sessions. The first session discusses the foundations and resource-aware data mining. The work *Towards Adjusting Mobile Devices to User's Behaviour* by Felix Jungermann, Katharina Morik, Nico Piatkowski, Olaf Spinczyk, Marco Stolpe and Peter Fricke 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.

The second session is concerned with ubiquitous and social applications and thus bridges the two general topics of the workshop and directly leads to the topic of the invited talk. In *A Framework for Mobile User Activity Logging*, Wofgang Woerndl, Alexander Manhardt and Vivian Prinz provide a unified approach for collecting user activity data on mobile devices for user modeling. This can be seen as a prerequisite for social approaches, e.g., as described in *Community Assessment using Evidence Networks* by Folke Mitzlaff, Martin Atzmueller, Dominik Benz, Andreas Hotho and Gerd Stumme. The paper presents an community assessment approach using evidence networks of user activities; the experiments indicate that (implicit) evidence networks are relatively well suited for community ratings. In *Exploring Country Level Gender Differences in the Context of Online Dating using Classification Trees*, 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.

The third session features the invited talk *Mining Social Context with Wearable Sensors* by Ciro Cattuto, in which he presents a platform for mining social context with wearable sensors, and illustrates it with several application studies.

The fourth session concludes the workshop with an outlook on the technology, its potential and limits. In *Predictability of Mobile Phone Associations* Bjørn Sand Jensen, Lars Kai Hansen, Jan Larsen, Jakob Eg Larsen and Kristian Jensen describe an analysis of lifelog data of sensors and mobile phones and discuss bounds for its predictability. The paper *Discovering Trend-Based Clusters 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.

We thank all participants of the workshop for their contributions and the organizers of the ECML PKDD 2010 conference for their support. Additionally, we want to thank our reviewers for their careful help in selecting and improving the provided submissions. Special thanks go to Harald Sack for his help in organizing an independent review process of a selected publication.

We are looking forward to a very exciting and interesting workshop.

Kassel, August 2010

Martin Atzmueller, Andreas Hotho

Towards Adjusting Mobile Devices to User's Behaviour

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

[1] Technical University of Dortmund,
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] Technical University of Dortmund,
Embedded System Software Group
Otto-Hahn-Strasse 16, Dortmund, Germany
olaf.spinczyk@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 [13] [22] [3]. 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 addressing sequences? We investigate the use of two extremes,

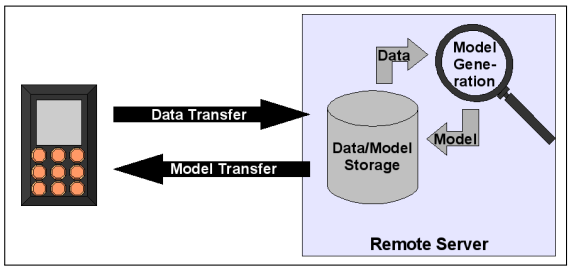


Fig. 1. Server-based Architecture

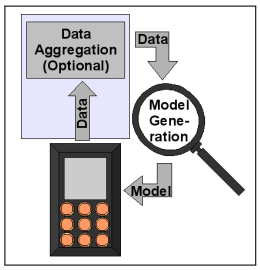


Fig. 2. Device-based Architecture

Conditional Random Fields (CRF) and Naive Bayes (NB). 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

In this section we present the first case-study, where log data are stored and analyzed on a server (data are described in Section 2.2). 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 2.3 and Section 2.4, respectively. The results are shown in Section 2.5.

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 for Access Prediction

We logged streams of system calls of type FILE, which consist of various typical sub-sequences, each starting with an `open`- and terminating with a `close`-call, like those shown in Figure 3. 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.

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 observations are compound to one sequence if they belong to the same process. The resulting dataset consists of 673887 observations in 80661 sequences, a snippet[3] is shown in Table 1.

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

```
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)

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

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

Table 2. Cost matrix

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

Table 3. Snippet of the final dataset using two features.

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.

2.3 Naive Bayes Classifier

The Naive Bayes classifier [10] 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:

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

Domingos and Pazzani [7] rewrite eq. (1) and define the *Simple Bayes Classifier* (SBC):

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

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

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

The term $p\left(x_1, x_2, ..., x_m\right)$ 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 $p\left(x_i|Y\right)$ for $i = 1 \ldots m$. 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 [7, 11, 8].

2.4 Linear-chain Conditional Random Fields

Linear-chain Conditional Random Fields, introduced by Lafferty et al. [12], 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

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

with the observation-sequence dependent normalization factor

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

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 [14], [19], [15] or stochastic gradient methods [25], [17], [18].

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

The actual class prediction for an unlabeled observation-sequence is done by the Viterbi algorithm known from Hidden Marcov Models [21], [16].

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 2.2.

2.5 Results of Server-based Prediction

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 (see Tables 4 and 6). CRF outperforms NB when using all features (see Tables 5 and 7). 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.

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 10 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*[4] for these experiments. Table 10 also shows the model sizes of CRF on both feature sets and various example set sizes.

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 4. Result of Naive Bayes Classifier on best two features, 10x10-fold cross-validated, accuracy: 94.45 ± 0.00, missclassification costs: 0.152 ± 0.001

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 5. Result of Naive Bayes Classifier on all 24 features, 10x10-fold cross-validated, accuracy: 91.84 ± 0.00, missclassification costs: 0.218 ± 0.002

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 6. Result of HMM-like CRF on the best two features, 10x10-fold cross-validated, accuracy: 95.49 ± 0.00, missclassification costs: 0.150 ± 0.000

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 7. Result of HMM-like CRF on all 24 features, 10x10-fold cross-validated, accuracy: 95.70 ± 0.00, missclassification costs: 0.143 ± 0.000

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 8. Result of linear-chain CRF on the best two features, 10x10-fold cross-validated, accuracy: 96.79 ± 0.00, missclassification costs: 0.112 ± 0.000

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	

Table 9. Result of linear-chain CRF on all 24 features, 10x10-fold cross-validated, accuracy: 96.89 ± 0.00, missclassification costs: 0.110 ± 0.000

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

#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 10. Storage needs (in kB) of the naive Bayes (nB) classifier model produced by *RapidMiner*, the HMM-like CRF (CRF++ (HMM)) and the linear-chain CRF (CRF++) 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

Table 11. Training time (in seconds) of the naive Bayes (nB) classifier model produced by *RapidMiner*, the HMM-like CRF (CRF++ (HMM)) and the linear-chain CRF (CRF++) on different numbers of sequences and attributes.

We used the open source CRF implementation *CRF++*[5] 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 11 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 [4], given a (multi)set S

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

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 [4]:

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:*

- *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 [5], 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 [6].

3.2 System Call Data for HHH

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	

Table 12. 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.

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. 12. The functional categorization of system calls into five groups is due to [20]. 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.

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. Each value originates from a taxonomy (*type*, *path* or *sequence*) that either can be derived dynamically from the data itself or has to be defined explicitly by the user. The groups introduced in Tab. 12 form the top level of the taxonomy for the hierarchical variable *type* (see Fig. 4). 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. 4). 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.

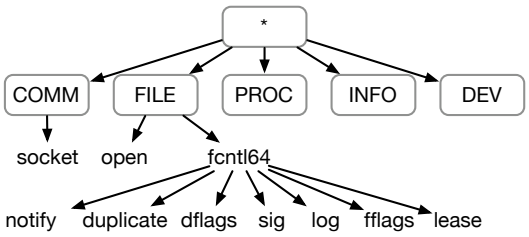


Fig. 4. Parts of the taxonomy we defined for the hierarchical variable *type*.

The *path* variable is filled whenever a system call accesses a file system path. Its hierarchy comes naturally along with the given path hierarchy of the file system. The *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.

We collected system call data from eleven applications (like Firefox, Epiphany, NEdit, XEmacs) with the *strace* tool (version 4.5.17) under Ubuntu Linux (kernel 2.6.26, 32 bit). All child processes were monitored by using option `-f` of *strace*. 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. 12, resulting in a whole of 110 log files comprising about 23 million of lines (1.8 GB).

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. Regarding run-time, all experiments were done on a machine with Intel Core 2 Duo E6300 processor with 2 GHz and 2 GB main memory.

		Memory			**Run-time**			**Similarity**	
		Min	Max	Avg	Min	Max	Avg	Avg	Dev
FA	T	19	151	111	16	219	79	**0.997**	0.006
	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
PA	T	7	105	70	15	219	74	**0.985**	0.010
	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

Table 13. 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 = *type* hierarchy, P = *path* hierarchy, S = *sequence* hierarchy).

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 13 shows the run-time and memory consumption of the Full Ancestry and Partial Ancestry algorithms using only the *type* hierarchy, the *type* and *path* hierarchy, or the *type*, *path*, and *sequence* hierarchy. Minimum, maximum 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.

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).

Figure 5 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.

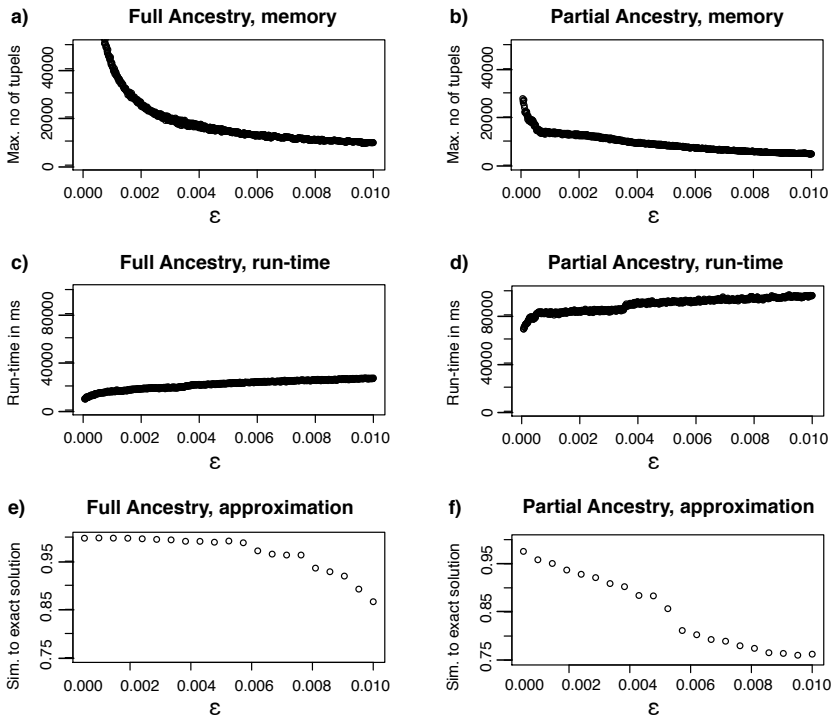


Fig. 5. Memory consumption (a, b), run-time (c, d) and similarity to exact solution (e, f) of HHH algorithms (three-dimensional) with varying ε, $\phi = 0.001$ on biggest log file of application Rhythmbox.

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

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

The so defined similarity measure is independent from the choice of ϕ, as no HHH sets need to be calculated.

The classification errors for different values of k, hierarchies and distance measures are shown in Tab. 14. 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.

	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

Table 14. Results for kNN (k = 3, 5, 7, 9), $\varepsilon = 0.0005, \phi = 0.002$ and distance measures DSM and TF, when only the *type* hierarchy or *type* and *sequence* hierarchy together are used.

4 Conclusion

Server-based and device-based learning has been investigated regarding resource constraints. Further experiments to measure resource consumption and prediction accuracy will be conducted on real mobile devices, like Android mobile phones, whose operating system is also based on 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) [9] and special kernel functions to measure the similarity of sequences [23]. 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 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 [24].

References

1. C. Bockermann, M. Apel, and M. Meier. Learning sql for database intrusion detection using context-sensitive modelling. In *Proc. 6th Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 196 – 205. Springer, 2009.
2. D. Bovet and M. Cesati. *Understanding the Linux Kernel, Third Edition*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2005.
3. B. M. Cantrill, M. W. Shapiro, and A. H. Leventhal. Dynamic instrumentation of production systems. In *Proc. of USENIX ATEC '04*, Berkeley, USA, 2004. USENIX.
4. G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava. Finding hierarchical heavy hitters in data streams. In *VLDB '2003: Proceedings of the 29th international conference on Very large data bases*, pages 464–475. VLDB Endowment, 2003.
5. G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava. Diamond in the rough: finding hierarchical heavy hitters in multi-dimensional data. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 155–166, New York, NY, USA, 2004. ACM.
6. G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava. Finding hierarchical heavy hitters in streaming data. *ACM Trans. Knowl. Discov. Data*, 1(4):1–48, 2008.
7. P. Domingos and M. Pazzani. Beyond independence: Conditions for the optimality of the simple bayesian classifier. In *Machine Learning*, pages 105–112. Morgan Kaufmann, 1996.

8. A. Frank and A. Asuncion. UCI machine learning repository, 2010.
9. K. Gupta, B. Nath, and K. Ramamohanarao. Conditional random fields for intrusion detection. In *21st Intl. Conf. on Adv. Information Netw. and Appl.*, pages 203–208, 2007.
10. T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, corrected edition, July 2003.
11. J. Huang, J. Lu, and L. C. X. Ling. Comparing naive bayes, decision trees, and svm with auc and accuracy. In *in: Third IEEE International Conference on Data Mining, ICDM 2003*, pages 553–556. IEEE Computer Society, 2003.
12. J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proc. 18th International Conf. on Machine Learning*, pages 282–289, 2001.
13. D. Lohmann, W. Hofer, W. Schröder-Preikschat, J. Streicher, and O. Spinczyk. CiAO: An aspect-oriented operating-system family for resource-constrained embedded systems. In *Proc. of USENIX ATEC*, Berkeley, USA, 2009. USENIX.
14. R. Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *COLING-02: proceedings of the 6th conference on Natural language learning*, pages 1–7, Morristown, NJ, USA, 2002. Association for Computational Linguistics.
15. J. Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, 1980.
16. L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb 1989.
17. N. N. Schraudolph and T. Graepel. Conjugate directions for stochastic gradient descent. In *ICANN '02: Proceedings of the International Conference on Artificial Neural Networks*, pages 1351–1358, London, UK, 2002. Springer-Verlag.
18. N. N. Schraudolph, J. Yu, and S. Günter. A stochastic quasi-Newton method for online convex optimization. In M. Meila and X. Shen, editors, *Proc. 11th Intl. Conf. Artificial Intelligence and Statistics (AIstats)*, volume 2, pages 433–444, San Juan, Puerto Rico, 2007. Society for Artificial Intelligence and Statistics.
19. F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 134–141, Morristown, NJ, USA, 2003. Association for Computational Linguistics.
20. A. Silberschatz, P. B. Galvin, and G. Gagne. *Operating System Concepts*. Wiley Publishing, 2010.
21. C. Sutton and A. McCallum. An Introduction to Conditional Random Fields for Relational Learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.
22. R. Tartler, D. Lohmann, W. Schröder-Preikschat, and O. Spinczyk. Dynamic AspectC++: Generic advice at any time. In *The 8th Int. Conf. on Software Methodologies, Tools and Techniques*, Prague. IOS Press. (to appear).
23. S. Tian, S. Mu, and C. Yin. Sequence-similarity kernels for SVMs to detect anomalies in system calls. *Neurocomput.*, 70(4–6):859–866, 2007.
24. C. Timm, A. Gelenberg, F. Weichert, and P. Marwedel. Reducing the Energy Consumption of Embedded Systems by Integrating General Purpose GPUs. Technical Report 829, Technische Universität Dortmund, Fakultät für Informatik, 2010.
25. S. V. N. Vishwanathan, N. N. Schraudolph, M. W. Schmidt, and K. P. Murphy. Accelerated training of conditional random fields with stochastic gradient methods. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 969–976, New York, NY, USA, 2006. ACM.

Bayesian Networks to Predict Data Mining Algorithm Behavior in Ubiquitous 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
{emenasalvas,seibe}@fi.upm.es
ysaygin@sabanciuniv.edu

Abstract. The growing demand of data mining services for ubiquitous environments motivates deployment of data mining algorithms that use context to adapt their behavior to present circumstances. Despite the efforts and results so far for efficient parameter tuning, there is a need to develop new mechanisms that integrate also the context information. Thus, in this paper, Bayesian networks are used to extract the effects of data mining algorithm parameters on the final model obtained, both in terms of efficiency and efficacy in a given situation. 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.

Key words: automatic data mining, data mining configuration, ubiquitous data mining

1 Introduction

Ubiquitous computing, still an immature computing paradigm, brings new challenges to software designers and also to designers of data mining software. In ubiquitous computing, processing takes place on the restricted resource devices that are embedded or spread in the environment which is subject to change all the time. This means that, ubiquitous computing paradigm implies lack of expert involvement on tuning the software where it is most needed due to scarcity of resources and variability of the context. Consequently, automatic configuration is required under changing context and resource constrained environments.

There is an increasing demand for intelligent applications on ubiquitous devices while data mining methods has been the main way to provide such intelligence. As an example, knowledge discovery and data mining can be an enabling

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

technology for more adaptive, dynamic, and autonomous social networking. Consider the scenario where mobile devices have the data mining ability for predicting the current activity and even the mood of the user. Combined with context information such as location and time enriched with more semantics, the mobile device can offer recommendations to the user taking into account the state of the other people in the social network of the user. For example if the user is at home but still not tired and in a jolly mood, the mobile device can communicate with the (mobile devices of the) friends of the user who are in the vicinity and in a similar mood to form an ad-hoc going-out group. Such an application will require mining of the data collected through various sensors to recognize the activity of the user and his/her mood taking into account the context as well.

Nevertheless, important challenges need to be addressed on the ubiquitous data mining design, which two of them will be focused on in this paper. First of all, the 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. Several 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 according to past behavior of the algorithm under a given situation.

Automatic parameter tuning research area has gained much interest in the recent years. A number of 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 satisfies best 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 prevailing 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 the ubiquitous environments, state of device and 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 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 and instantiates it for a case. In Section 3, we explain the experiment that we performed in order to validate of the proposed approach. Finally, future work is described in Section 4.

2 Proposed Approach

We present a mechanism to determine the algorithm configuration in a resource-aware and context-aware manner with respect to a data mining request issued. 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

The main objective can be stated as "to determine automatically the configuration of a data mining algorithm which will run on an ubiquitous device". This objective requires understanding the elements influencing the solution:

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

In addition, the most important issue is how to change or improve configuration setting decisions as the circumstances change. The decisions must be adaptable to changing conditions as a data miner expert adapts his decisions when the conditions change.

2.2 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 structure that shows the conditional dependencies between domain variables and may also be used to illustrate graphically the probabilistic causal relationships among domain variables. A Bayesian network consists of a directed acyclic graph and probability tables. The nodes of the network represent the domain variables and an arc between two nodes (parent and child) indicates the existence of a causal relationship or dependency among these two nodes. Associated with each node there exist a probability table (PT). If the node has no parents, its probability table contains the prior probabilities else the conditional probabilities between the node and its parents. Although the domain variables can be continuous, they are discretized most of the time for simplicity and efficiency. Besides representing the dependencies between domain variables, a Bayesian network is used for inferencing the probability of a variable given the observations of other variables. In depth knowledge on Bayesian networks can be found in [19].

Learning the Bayesian network structure from data rather than drawing 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 an ubiquitous environment. A discussion of the literature can be found in [5].

2.3 Basis of the Approach

We propose machine learning as the main mechanism to learn from past executions of data mining algorithms. Without a loss of generality, we selected Apriori [1] as the prototype algorithm for automatic configuration but the proposed mechanism is also applicable to any other algorithm. In particular, we propose to construct a Bayesian Belief Network using the information collected during algorithm's previous executions in order to predict future behaviors. The content of the information is determined by considering the elements that influence the solution of the problem (given in subsection 2.1). This information which is recorded as history of execution records is divided into three groups:

- Circumstantial: Information about the conditions of the resources and the context states that is obtained prior to execution of the algorithm.
- Configuration Parameters: The values that the algorithm parameters receive.
- Quality Measurements: Efficiency and efficacy related information. They include resource consumption measurements such as duration of the data mining, average memory used or indicators of model quality such as minimum support or confidence of an association rule model.

Groups are determined by taking into account the relationships that we want to examine by building a Bayesian network. Moreover, the content of the groups cover all the elements given in subsection 2.1 except the ones related to the data to be mined as we focus on the ubiquitous aspect in this work.

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

- Association rules are needed for a specific data set,
- Current circumstance and the quality requirements of the data model are determined to be used for the configuration setting,
- The algorithm which will discover the association rules is configured autonomously by inferencing from the Bayesian Belief Network that represents circumstances, parameters and quality measurements,

In the next subsections we describe in more detail the process of building the Bayesian network.

2.4 Definitions

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

Circumstance: Circumstance 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 Criteria: Quality criteria is defined by a set of ordered pairs *(q,l)* where q is a quality measurement and l is the required level for this measurement. Quality measurements are metrics of efficiency or efficacy of the algorithm. Quality criteria define either the expected efficiency of data mining or efficacy of the model or both under the given circumstances.

2.5 Mechanism to Predict Ubiquitous Data Mining Configuration

We propose to build a Bayesian network using data collected from past executions of the data mining algorithm. The Bayesian network reflects the conditional dependency between the fields of execution records. Configuration is inferred from the Bayesian network. More specifically, the most likely configuration is determined from the Bayesian network by estimating the probabilities of possible configuration settings from previous Apriori runs in execution circumstances similar to current in order to obtain quality levels similar to the required.

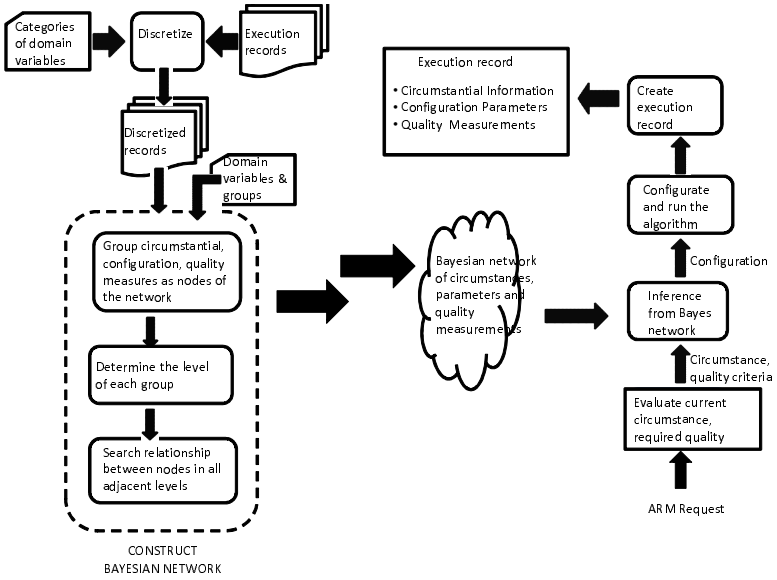


Fig. 1. Bayesian Network construction steps and data mining configuration mechanism.

We used the K2 algorithm proposed by Cooper and Herskovits([8]) as the basis for constructing Bayesian network. Before building the Bayesian network, we preprocessed the historical information and discretized the fields of execution records. 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. In our case, we have three groups of variables. The relationships among the variables within a group are not interesting. For example, we are not interested in the relationship among circumstantial variables such as *location* and *memory available*. For this reason, we modified the K2 algorithm accordingly and looked for relationships among nodes belonging to different groups of variables. Furthermore, we assigned a level to each group based on the possible cause-effect relationship between them and we used the levels to prevent the nodes in the lower level groups to be the parents of the

nodes in the upper level groups. Fig. 1 illustrates Bayesian network construction steps that we propose.

The Bayesian network that we construct from historical data represents the probabilistic relationship between circumstance states, discretized possible parameter settings, and measured as well as discretized quality measurements. Appropriate setting of an algorithm's parameter is extracted from the Bayesian network.

2.6 Instantiation of the Approach for Apriori

In this section, automatic configuration setting is instantiated for the well known association rule mining algorithm Apriori. A possible instantiation of the approach is shown by assignments made to the parameters of Apriori, circumstances and quality criteria.

Parameters. We use the parameters for the implementation of Apriori available by Weka software ([13]). The list of parameters that we selected for tuning are as follows:

- upperBoundMinSupport: upper bound for minimum support.
- delta: the factor which minimum support is decreased each time Apriori is iterated.
- lowerBoundMinSupport: lower bound for minimum support.
- numRules: number of strong association rules.
- minMetric: minimum confidence.

A possible configuration that is obtained as a result of inferences made from the Bayesian network, is as follows:
Configuration = {(upperBoundMinSupport, 0.9), (lowerBoundMinSupport, 0.5), (delta, 0.05),(numRules, 15),(minMetric, 0.9)}

Circumstance. We restrict the instantiation of circumstance to a small number of resources and context. Resources that we use are *memory* and *CPU*. Context features considered in the instantiation are *location* and *time*. We discretized the actual figures that we obtained for the availability measures and converted to states. We define five states for the resource availability measures numbered from one to five where small numbered states indicate scarcity of the resource; high numbered ones, the opposite. *Location* refers to physical location and time slots of the day are used when discretizing *time*. Some possible instantiations of circumstances are as follows:
Circumstance 1= {(memory available, 4), (location, home)}
Circumstance 2= {(CPU available, 1),(memory available, 2),(location, office)}

Quality Criteria. Quality measurements selected for the instantiation are divided into two in relation to the kind of criteria they measure (Table 1). The ones for measuring efficiency can be obtained by calling system primitives. On

the other hand, it is possible to extract the efficacy related ones from the data mining model generated. Some possible instantiations of quality criteria are as follows:
Quality Criteria 1 ={(average memory usage, 1)}
Quality Criteria 2 ={(total CPU time, 1), (number of rules discovered, 3)}

Table 1. Quality measurements for efficiency and efficacy.

Efficiency	Efficacy
Average memory usage	Minimum support of the resulting model
Maximum memory usage	Number of iterations
Total CPU time	Number of rules discovered
Total number of CPU cycles	Minimum confidence obtained
Duration	

3 Experimental Evaluation

We proposed an automatic parameter setting mechanism in which the decisions are based on Bayesian network of data mining algorithm's past executions. Our aim in conducting experiments is to validate that it is possible to establish the configuration of the algorithm by making use of the proposed mechanism. In order to do that, we created Bayesian network of Apriori executions to be used for understanding the behavior of Apriori. Besides, we employed another approach, full factorial experiment design to compare and verify the results obtained from the Bayesian network. In full factorial experiment design, we examined the effects of the parameter settings of the algorithm against a quality metric. Same data used for the Bayesian network construction is supplied to the full factorial experiment design. Finally, we compared the inferences made from the Bayesian network against the results of the full factorial experiment design. There should

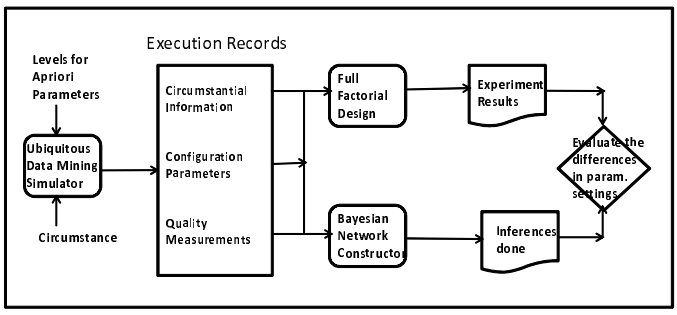


Fig. 2. Experiment phases.

be a sufficient number of Apriori executions in order to obtain a sound and reliable Bayesian network. Considering this requirement, we preferred to use full factorial design where all combinations of parameters for the determined levels are utilized during the tests. We divide the experiment into three phases:

1. Multi-level full factorial design is used to reveal the factors (parameters) that are effective on the quality indicators under a given circumstance.
2. Bayesian network is constructed in order to infer the probabilistic relationships among parameters, circumstances and quality criteria.
3. The accuracy of the proposed mechanism is assessed by comparing the results of two phases.

In order to generate historical data, we used an ubiquitous data mining simulator. Fig. 2 shows the interaction of the experiment phases and the ubiquitous data mining simulator.

3.1 Ubiquitous Data Mining Simulator and Experiment Data

We have developed a simulator in order to incorporate the features of ubiquitous data mining that we are interested in while creating records of Apriori executions for the experiment. While running Apriori, a possible circumstance consisting of resources states and context is given as input to simulator for each Apriori execution. An execution environment according to the circumstance provided is simulated. For example, if the stated resource state is the scarcity of memory, the simulator starts dummy processes to use up the memory in order to run Apriori in a memory constrained situation. In the same way, information given about context is stored in the execution record as part of circumstantial information. Briefly, the data mining simulator reads circumstance and Apriori parameters from a file, generates the resource scarcity conditions if the given circumstance requires and runs Apriori by calling Weka with the given parameters. Upon completion, an execution record is written consisting of input context fields, fields showing resources availability, Apriori parameters, resource usage fields and the data mining model.

While generating possible parameter values of Apriori, all combinations of determined levels of process factors are considered as in full factorial design. Full factorial design is one of the Design of Experiment (DoE) methods [17] which is statistically determining the effects of factors of a process to its response by systematically varying the levels of 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. In our case, Apriori parameters correspond to full factorial experiment design *factors* and possible settings of Apriori parameters correspond to *levels* in factorial design terminology. Hence, we determined possible settings for Apriori parameters (Table 2) and run for all combinations of levels.

We simulate an ubiquitous device by composing different availability of resource and context. Therefore we determined two groups of levels to be used

Table 2. Levels used for parameters.

Circumstance	Mnemonic	Parameter	Levels
Group 1	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
Group 2	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

on different circumstances (Table 2). We run Apriori for every combination of levels so the number of distinct settings in the groups are 2250(3x6x5x5x5) and 180(3x3x5x2x2) respectively. The states of the context and resource features that we used in forming the circumstances are {*home*, *office*} and {*short on memory*, *cpu bottleneck*, *none*} respectively. All the resource states are simulated for each context state, resulting in six circumstances. Each group is used for one context state as in the first group for *home* and the second group for *office*.

3.2 Multi-level Full-Factorial Experiment Design

In this phase, we apply multi-level full factorial design using Apriori execution records generated in the manner explained in the previous section. Applying multi-level full factorial design to the history of Apriori execution data determines which Apriori parameters (factors) effect which quality measurement (response). Since we run Apriori by simulating specific circumstances, we are able to analyze the effects for each circumstance. We obtained an "effects table" of quality measurements by parameters for each circumstance.

We used experiment software Minitab([16]) to calculate the estimated effects and to plot the analysis results. In this paper, we limited our discussion of results to six quality measurements: *average memory use*, *total CPU time*, *duration*, *maximum memory use*, *total CPU cycles*, *minimum support of the model*. We determined the effects of five Apriori parameters to each of the quality measurement under a single, different circumstance. For example, when *home-memory low*, we examined the effects of five Apriori parameters to *average memory use* or when *office-CPU bottleneck*, the effects of parameters to *total CPU cycles* and so on.

Fig. 3 illustrates the full factorial 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. 3, 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

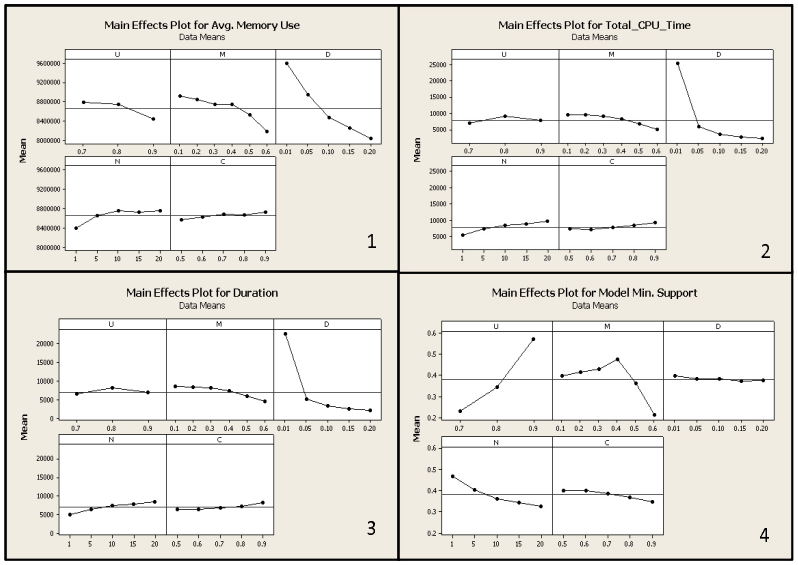


Fig. 3. Main effects plot of 4 quality measurements for *home-short on memory.*

this parameter is effective on the measured value. For example, we analyze the main effects plot for *average memory use* in quadrant 1 and we observe that D is most effective on *average memory use* since varying its value causes big differences on the mean *average memory use*. The plots of means give an insight on understanding the effect of a certain parameter to a quality measure. On the other hand, we considered the F test values that are supplied by Minitab([16]) to determine the significance of the effect.

The next step is to determine the appropriate value of the parameter which is designated as effective on the measured criteria. We choose the value that has the smallest mean of response for its factor level combinations as the appropriate value. We present the results of full factorial experiment design in Table 3 where we compare against the results of the Bayesian network.

3.3 Parameter Setting by Bayesian Network Inferences

In this phase, we apply our mechanism and obtain the results, that is, the parameter settings of Apriori from the Bayesian network. First, we explain in detail our considerations while constructing the Bayesian network before presenting the results of this phase.

Execution records generated by ubiquitous data mining simulator are first discretized, then they are used to construct the Bayesian network (Fig. 4). We made use of the K2 algorithm ([8]) but we grouped the nodes while constructing the network and searched for causal relationship among these groups of nodes.

The nodes in the upper level represent the circumstance, middle level nodes represent Apriori parameters, and finally the lowest level nodes are quality measures.

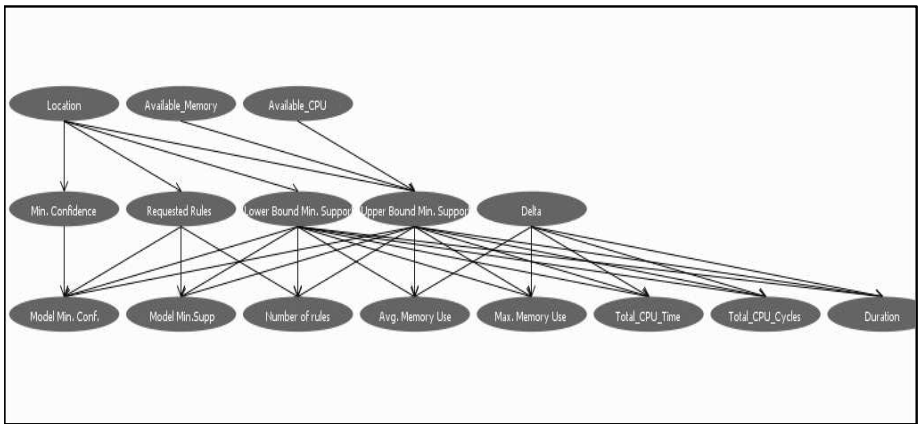


Fig. 4. Bayesian network of Apriori runs.

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 metrics show which parameters are effective on which quality measures.

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 is not meaningful. We assumed each circumstance variable 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 of parameters by inferencing from the Bayesian network given in Fig. 4. In particular, we evaluated p(x|E) which is the conditional probability of x (parameter variable) given E (circumstantial and quality measure variables). A pseudo code of this calculation is given below. We estimate from previous Apriori runs, the support for assigning a certain value to a parameter when execution circumstances similar to current and quality levels similar to the required were observed. Given the circumstances and quality

requirements, we repeat the estimation of the most likely assignment for every parameter of the algorithm.

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 criteria set of C_k
     where (q_kj,v_kj) is a quality measure, 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 E 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 | E )
          p_xy with highest Pr is the appropriate setting of P_x for C_k.
```

The parameter settings that we obtained by applying the pseudo code above to the Bayesian network given in Figure 4 are presented in Table 3 and are compared against the parameter settings of full factorial design.

3.4 Comparison of Results

We propose to use Bayesian network for automatizing the parameter tuning of data mining algorithms. We discovered the relationships among circumstantial variables, parameters and quality measures to use this information for probabilistically estimating the appropriate parameter settings. In order to validate the results that are obtained from the Bayesian network, we used another approach, full factorial design to achieve the same goal. In full factorial design, regression is used to determine the effect of a parameter to a quality measure. Both of the approaches can be used to determine the parameter settings of an algorithm where the outcomes of Bayesian network and full factorial design 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

In factorial design, the effects of factors can be analyzed against a single response at a time. Although, this restriction does not apply to Bayesian inferences, we used a single quality measure in order to analyze and compare the results of the Bayesian inference with the full factorial design results. The flexibility of using multiple variables on forming the evidences of inferences is one of the strengths of Bayesian network over full factorial design. We simulated a specific circumstance each time we analyzed the effects of parameters onto a single quality measure. In both approaches, we aimed to estimate the appropriate values for five parameters considering six circumstance-quality measures. Table 3 shows the results we obtained from both approaches. *Minimum confidence* is not included in Table 3, since it is observed that *minimum confidence* is not related to any of the considered quality measures in both of the approaches. The first parameter value given on each column is obtained by inferring from the Bayesian network whereas the second one is from the full factorial design (B/F).

Table 3. Parameter settings by circumstance.

Circumstance	Quality Measure		U	M	D	N
home-short on memory	average memory usage	B/F	0.9/0.9	0.6/0.6	0.2/0.2	
home-CPU bottleneck	total CPU time	B/F	0.7/-	0.6/0.6	0.2/0.2	-/1
home-no constraints	model's minimum support	B/F	0.7/0.7	0.6/0.6		20/20
office-short on memory	maximum memory usage	B/F	0.9/0.7	0.6/0.6	0.2/0.2	
office-CPU bottleneck	total CPU cycles	B/F	0.7/0.7	0.6/0.6	0.2/0.2	
office-no constraints	duration	B/F	0.7/0.7	0.6/0.6	0.2/0.2	

It is possible to say based on the results in Table 3, that in majority of the cases, parameters that are found to have effect 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 Conclusion

Anticipating the importance of autonomous and adaptable behavior incorporation in ubiquitous data mining enabled us to propose Bayesian network for understanding the algorithm behavior and determining the appropriate parameters. Considering the characteristics of ubiquitous environments, we stressed the usage of circumstance on determining appropriate parameter values. We also aimed at recommending the parameter settings that most likely satisfies the required quality. 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.

As the result of our simulation experiment, satisfactory parameter and quality measure relationships to recommend parameter settings, are 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 optimal setting obtained from full factor analysis which is a completely different approach.

In the future, we will assess the adaptability of the proposed approach by conducting experiments using recommended parameter values obtained in this work. We aim to extend this work by constructing Bayesian network structures for alternative variety of circumstantial variables and quality measures. Assessing the accuracy and the cost of the estimation and analyzing when to update the network are still the open issues in which we also plan to work in the near future.

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. and Srikant R.: Fast Algorithms for Mining Association Rules. In : Proceedings of the Int. Conf. on Very Large Data Bases (VLDB'94), pp. 487–499. Morgan Kaufmann, San Francisco (1994)
2. Amstrup, S. C., Marcot, B. G., and Douglas, D. C.: A Bayesian Network Modeling Approach to Forecasting the 21st Century Worldwide Status of Polar Bears. In : Arctic Sea Ice Decline: Observations, Projections, Mechanisms, and Implications. Geophysical Monograph 180, 487–499. American Geophysical Union, Washington, DC (2008)
3. Beinlich, I.A., Suermondt, H.J., Chavez, R.M., and 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, pp. 247–256. London (1989)
4. Birattari, M., Stutzle, T., Paquete,L. and Varrentrapp, K.: A Racing Algorithm for Configuring Metaheuristics. In : GECCO '02 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. and Mitkas, P.A.: Agent Mining: The Synergy of Agents and Data Mining. IEEE Intelligent Systems, 24, 64–72, (2009)
7. Charniak, E., and 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, pp. 1074-1079. Menlo Park, California (1989)
8. Cooper, G. F. and 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. and Laguna, M.: Fine-Tuning of Algorithms Using Fractional Experimental Designs and Local Search. Oper. Res. 54, 99-114 (2006)
10. Gagliolo, M. and Schmidhuber, J.: Learning Dynamic Algorithm Portfolios. Annals of Mathematics and Artificial Intelligence 47, 295-328 (2006)
11. Gaber, M.M. and 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, NY (2006)
12. Haghighi, P.D., Zaslavsky, A., Krishnaswamy, S., and 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. and Witten, I.H.: The WEKA Data Mining Software: An Update. SIGKDD Explorations, 11, (2009)
14. Hood, and C., Ji, C.: Proactive Network Fault Detection. In: Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution, INFOCOM, pp. 1147. IEEE Computer Society, Washington, DC (1997)
15. Hutter, F., Hoos, H. H., and Stutzle, T.: Automatic Algorithm Configuration Based on Local Search. In: 22nd National Conference on Artificial Intelligence, pp. 1152–1157. AAAI Press, (2007)
16. Minitab Inc., `http://www.minitab.com/en-US/`
17. Montgomery, D.C.: Design and Analysis of Experiments. John Wiley and Sons, (2006)
18. Pavon, R., Diaz, F., Laza, R., and 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, (1988)
20. Srivastava, B. and 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 (2005)

A Framework for Mobile User Activity Logging

Wolfgang Woerndl, Alexander Manhardt, Vivian Prinz

TU Muenchen, Chair for Applied Informatics / Cooperative Systems (AICOS)
Boltzmannstr. 3, 85748 Garching, Germany
{woerndl, manhardt, 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.

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 (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.

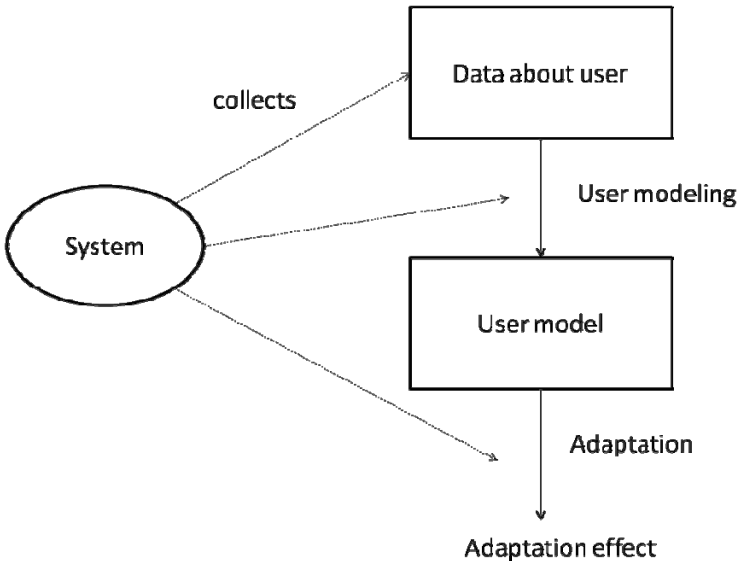


Fig. 1. User modeling process [1]

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 a 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.

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. Finally, we give a summary and outlook for future work in Section 5.

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.

MobSens is a system to derive sensing modalities on smart mobile phones [7]. 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 [8]. 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 [8].

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.

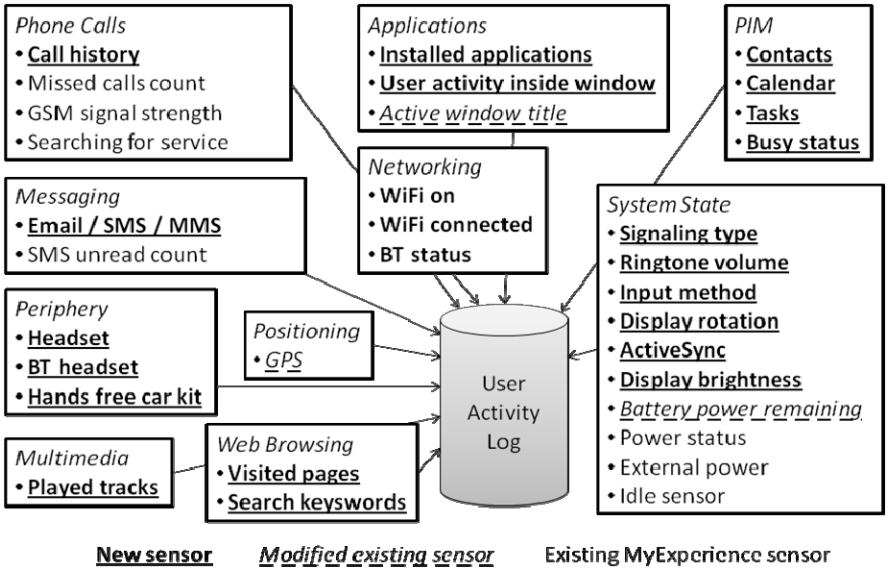


Fig. 2. Available sensors

MyExperience allows for configuring sensors via an XML file [8]. 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.

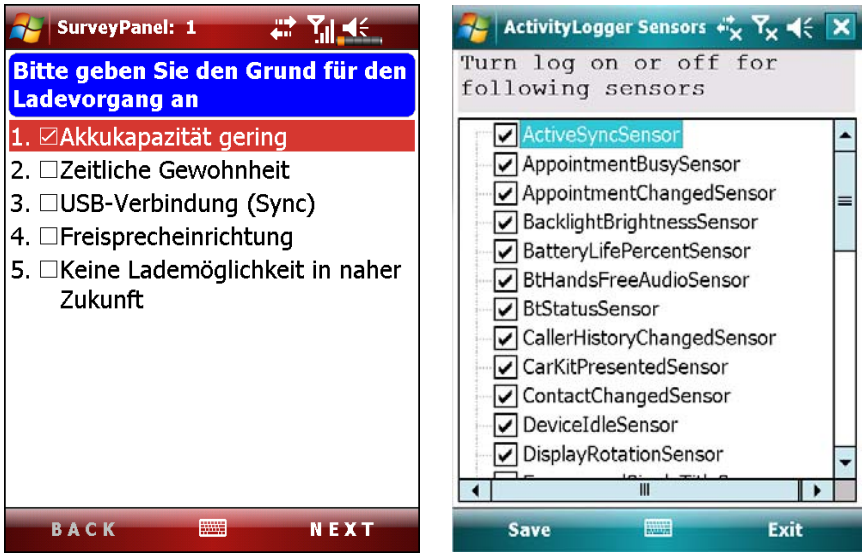


Fig. 3. Requesting explicit user feedback (left), and selecting sensors (right)

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.

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 file 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 straight-forward to analyze where the user has performed certain actions. This is also included in the following evaluation section.

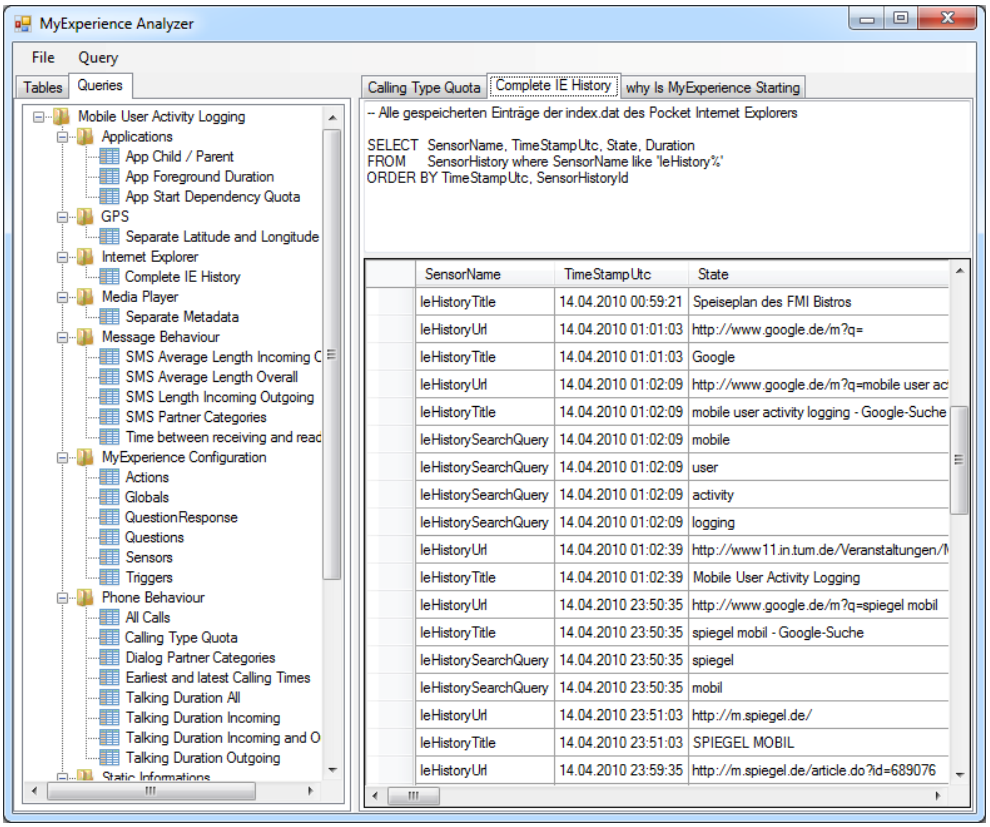


Fig. 5. Analyzer screenshot

4. Evaluation

In this section, we explain the evaluation of our approach. Note that we have focused on the collection of user data only at this time. 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. 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 work day. 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.

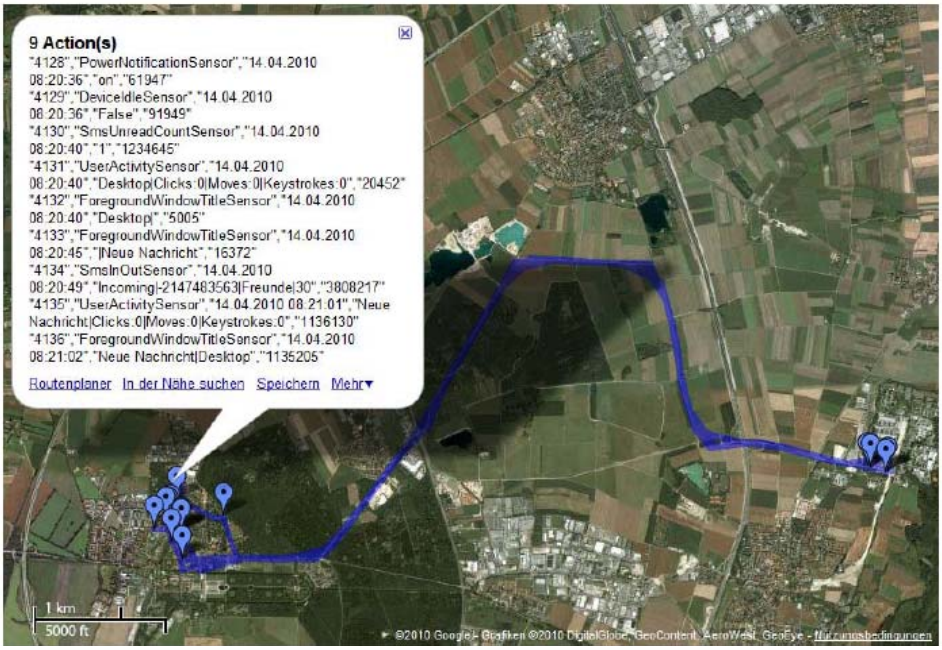


Fig. 6. Visualization of log data

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

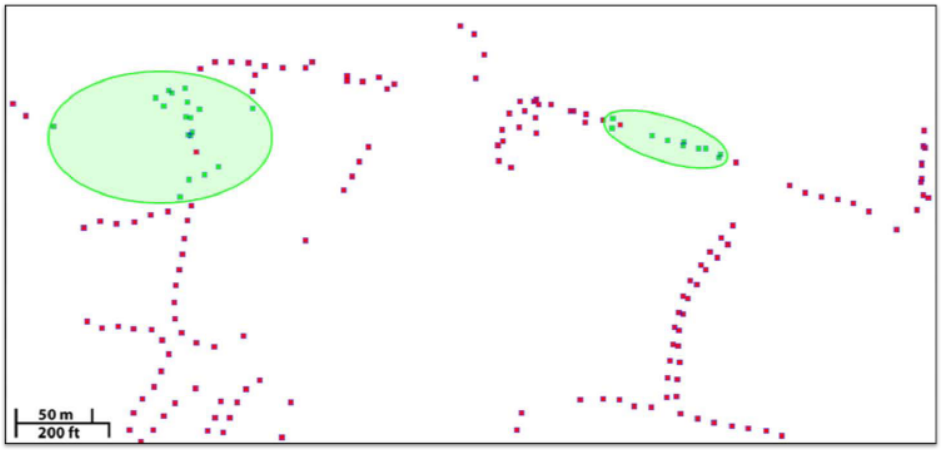


Fig. 7. WLAN activation based on position

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.

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. We have not investigated the actual data mining methods needed to identify such patterns so far. Yet our tests showed that our mobile user activity logger produced data in appropriate granularity for user modeling.

5. Conclusion and Outlook

The goal of this work is a unified approach for collecting data about user actions on mobile devices in a 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) [9] and the Contextualized Attention Metadata framework (CAMf) [10].

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). 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.

References

1. Brusilovsky, P., Maybury, M.T.: From Adaptive Hypermedia to the Adaptive Web. Communications of the ACM, vol. 45, no, 5, pp. 30-33 (2002)
2. Subramanya, S.R., Yi, B.K.: Enhancing the User Experience in Mobile Phones. IEEE Computer, vol. 40, no. 12, pp. 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 Wild Web (WWW 2009), Madrid, Spain, ACM Press, pp. 791-800 (2009)
6. Choudhury, T. et.al.: The Mobile Sensing Platform: An Embedded Activity Recognition System . IEEE Pervasive Computing, vol. 7, no. 2, pp. 32-41 (2008)
7. Kanjo, E., Bacon, J., Roberts, D., Landshoff, P.: MobSens: Making Smart Phones Smarter. IEEE Pervasive Computing, vol. 8, no. 4, pp. 50-57 (2009)
8. MyExperience project web site. http://myexperience.sourceforge.net/. Accessed June 2010.
9. APML web site. http://apml.areyoupayingattention.com/. Accessed June 2010.
10. CAMf web site. http://www.ariadne-eu.org/index.php?option=com_content&task=view&id=39&Itemid=55. Accessed June 2010.

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 evidence networks using 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 evidence networks using 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

Social applications and social networks provide a wealth of data that can be utilized for improving the user experience of the system. Appropriate recommendations for the users, for example, are an important criterion. Then, a peer group, or community of users similar to the targeted user is often a helpful resource for automatic approaches. In order to identify communities, *community mining* and *community detection* methods are applied, in order to identify groups of users which share a common interest or expertise.

While there are a lot of prominent methods for community detection, e.g., [18, 19, 9], the resulting models need to be assessed and evaluated. However, often there is no evaluation data at hand in order to evaluate the discovered groups comprehensively. Usually only one data source is available, for example, relational data on user–resource information, or also link data between users. Therefore, an accurate effective evaluation is non-trivial, since reliable (secondary) evaluation data is sparse or non-existent.

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 now become available that 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.

This paper proposes an approach for the evaluation of communities using implicit information formalized in so-called 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.

Considering our own system BibSonomy[3][3] as an example, we distinguish explicit and implicit relations. The friend graph, for example, indicates explicit friendship relations. Then, these graphs directly indicate communities according to the link structure. Implicit relation networks capture the implicit relations, that is, links derived from user behavior, e.g., visiting a page, clicking on a link, or copying a resource. Explicit networks are usually sparse and small and often only capture the characteristics of selected communities. In this respect, implicit networks capture more information and can be used for an evaluation directly, or for complementing explicit networks [22]. Furthermore, using implicit information captured by user actions and behavior is usually more cost-effective than starting expensive user-studies. We introduce several (implicit) evidence networks and discuss their features. Additionally, we present a comprehensive evaluation using user data from the BibSonomy system.

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.

The rest of the paper is structured as follows: Section 2 introduces the proposed approach for community evaluation using evidence networks. It outlines the basic notions of the approach, and discusses evidence networks and their characteristics. After that, we analyze and compare in Section 3 the features of the networks using data from the real-world BibSonomy system. Finally, Section 4 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., [9]. 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.

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

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* $A_{ij}, i = 1 \ldots n, j = 1 \ldots n$ 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 $v_0, \ldots, v_n$ of nodes with $n \geq 1$ and $(v_i, v_{i+1}) \in E$ for $i = 0, \ldots, n-1$. 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 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.

A community is intuitively defined as a set of nodes that has more and/or better links between its members than with the rest of the network. Formally, communities can be defined using certain criteria, for example, edge counts within a community compared to the edge counts outside, cf. [14]. The criteria are formalized using quality measures for communities. There are a variety of measures for community analysis, cf. [14]. In the context of evaluation measures for evidence networks we consider two measures: *Conductance* and *Modularity*. 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. [14]. 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

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

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

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

where k_i^{in} and k_j^{out} are i and j's in- and out- degree respectively [13].

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 user-graph structure, for example, into a click-graph, a visit-graph, or into a copy-graph of users.

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 typically also found in other resource sharing and social applications. Even more implicit user interaction occur 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 [23], 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, [11]). 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., [10] for a survey). These indices capture the intuition of internally densely connected clusters with sparse connections between the different clusters. Furthermore, based on the analysis of several social networks, Newman defines the modularity measure [18]: It 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 [14]. 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 [22] 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 (implicit) evidence networks: 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., [17]. 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 [17], 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.

2.4 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 evaluation measures, metrics and evaluation paradigms.

Karamolegkos et al. [11] propose metrics for assessing user relatedness and community structure by considering user profile overlap. They evaluate their metrics in a live setting, focussing on the optimization of the given metrics. Using a metric which is purely based on the structure of graphs, Newman presents algorithms for finding communities and assessing community structure(s) [19]. 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 [14].

Recently Siersdorfer et al. [22] 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. [21] investigated the relationship of topological closeness (in terms of the length of shortest paths) with respect to the semantic similarity between the users.

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. [2] 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. [12] 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 [6]. Mislove et al. [16] 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 [20].

3 Evaluation

In the following, we first describe the data used for the evaluation of the evidence networks. We used publicly available data from the social bookmark and resource sharing system BibSonomy. After that, we describe the characteristics of the applied evidence networks, and present the conducted experiments. We conclude with a detailed discussion of the experimental results.

3.1 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[4]. 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.

	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

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

[4] Note: For privacy reasons a user may deactivate this feature!

3.2 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. [16] 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 [8], that is $p(k) \sim k^{-\alpha}$ where $\alpha > 1$ is the exponent of the distribution. Online social networks [16], collaborative tagging systems [7], scientific collaboration networks [1] among others are shown to expose power law distributions.

For comparability, we calculated a best fitting power law model for each distribution using a maximum likelihood estimator [8] 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.

	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

Table 2. Power law parameters

3.3 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 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* [15]. 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

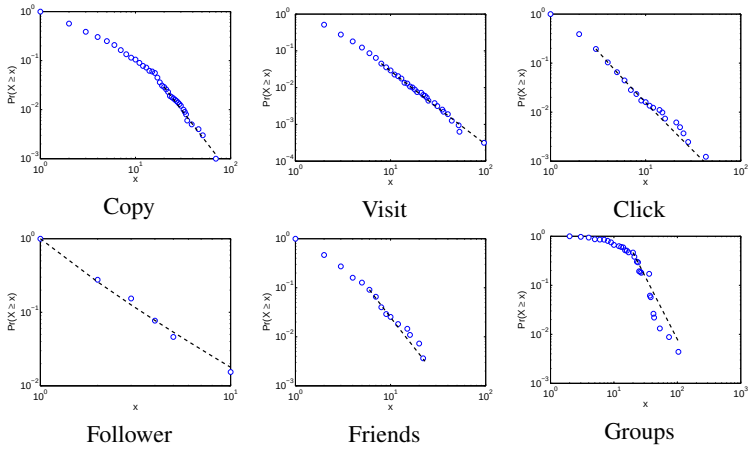


Fig. 1. In-degree distribution of the different evidence networks

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., [4] 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* [5] method for building topics, which efficiently build interpretable tag clusters and has been successfully applied in similar contexts to tagging systems (cf. [22]). 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.

3.4 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.

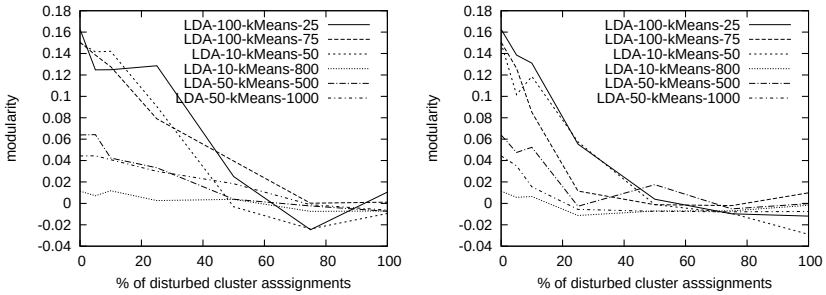
We started with community structures constructed by the basic feature clustering described above, using 10, 50, 100, and 500 topics, and constructing clusterings ranging from 10 to 1,000 clusters in total. Any clustering or community detection method

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.

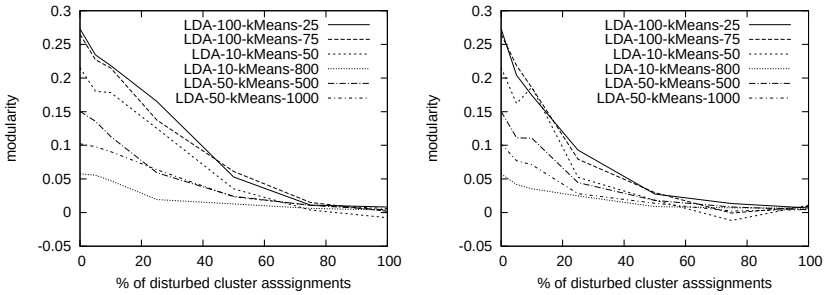
Figures 2 and 3 show the corresponding results of calculating the modularity for each evidence network at every level of disturbance in the underlying community structure (higher modularity values indicate stronger community structure). Similarly, Figures 4 and 5 show the results of calculating the conductance. 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 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 cluster size distributions). This behavior is consistent with the corresponding bias of the applied measures as discussed in [14].

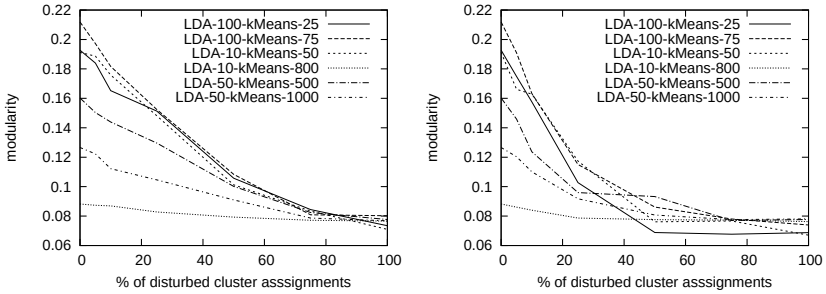
The preceding results consider the different evidence networks independently. However, we ultimately want to use the implicit networks as supplement for the sparse explicit social structures (in particular the Friend-Graph). We therefore expect the assessment of community structures applying the implicit 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 of the explicit networks. Table 3 shows the corresponding correlation coefficients for 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 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



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



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



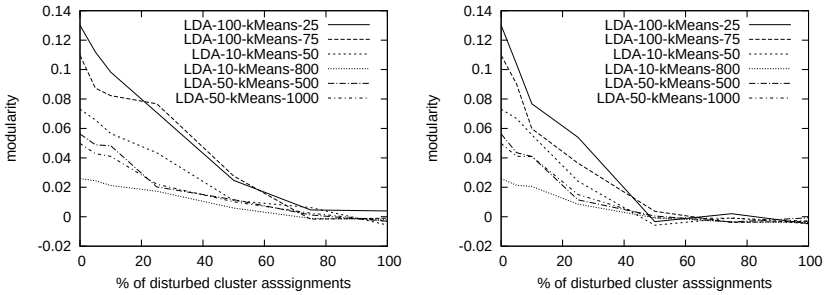
(c) Group-Graph (randomized left, shuffled right).

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

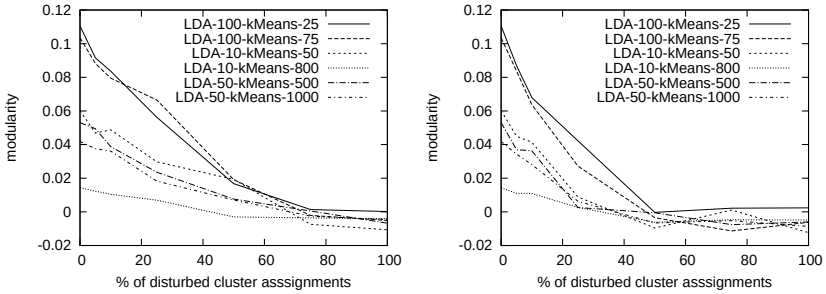
with low standard deviations. In comparison, repeating the same experiment with the group graph as the most dense existing social structure shows lower correlation coefficients with higher standard deviation, cf. Table 4.

3.5 Discussion

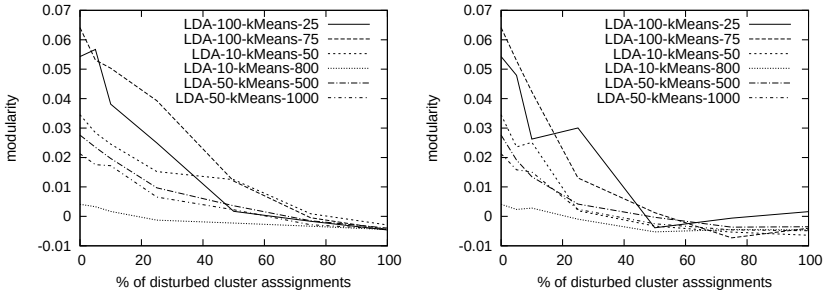
The experimental results presented in the previous section indicate that implicit evidence networks used for assessing the quality of a community structure are surprisingly



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



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

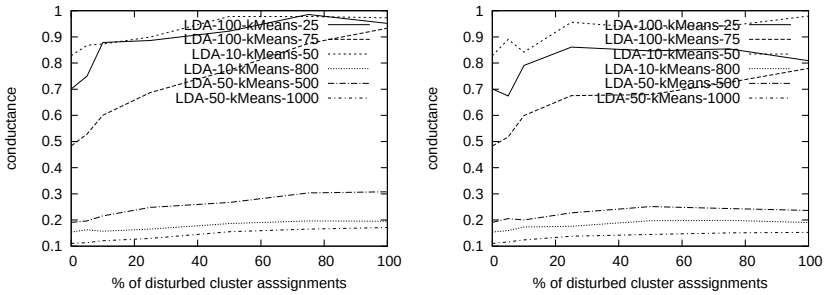


(c) Visit-Graph (randomized left, shuffled right).

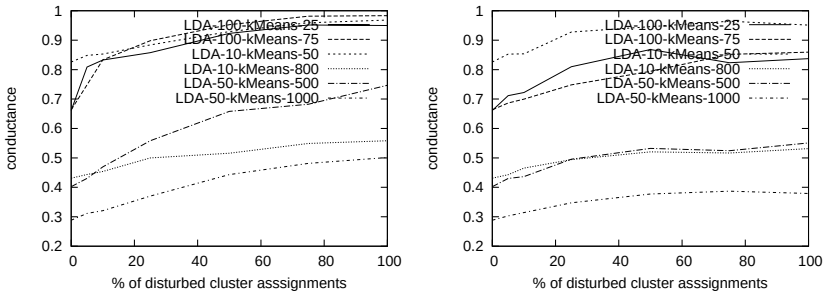
Fig. 3. Modularity calculated on different clusterings at varying levels of disturbed cluster assignments relative to implicit evidence networks

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.

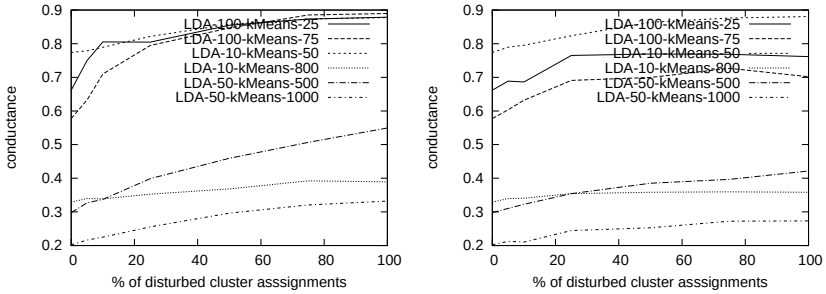
The implicit networks show a lower correlation with the group graph. At the first glance, this looks like a disappointing result. But the analysis of the group graph shows, that its properties significantly differ from typical social networks as discussed in [17, 16]. Most strikingly, its degree distribution follows not a power law and its distribution



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



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

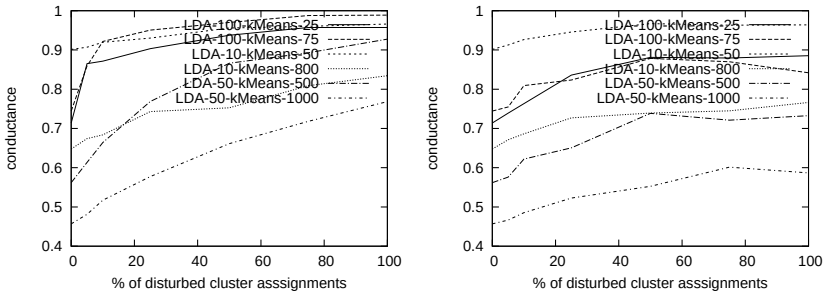


(c) Group-Graph (randomized left, shuffled right).

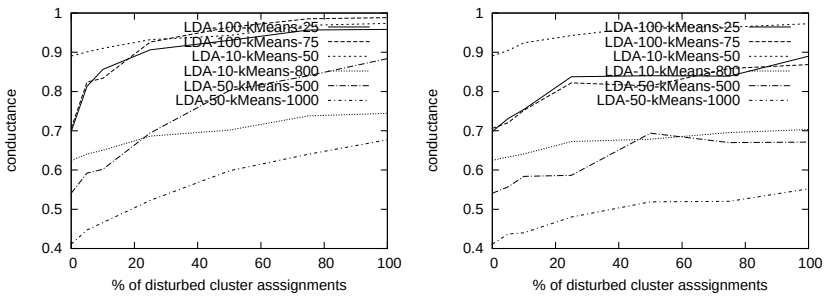
Fig. 4. Conductance calculated on different clusterings at varying levels of disturbed cluster assignments relative to explicit evidence networks

of 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.

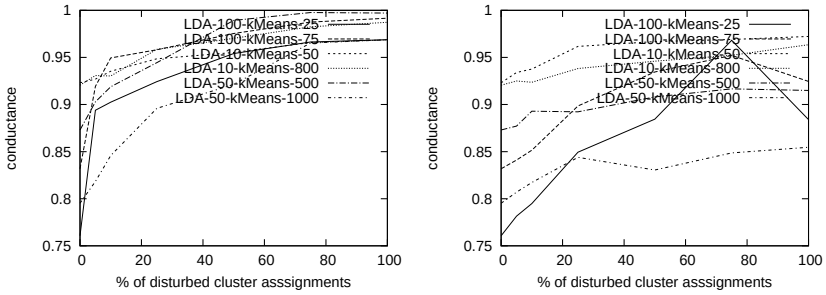
Furthermore, as exemplary shown in Figures 6, we observe in our experiments that known biases of the considered quality measures [14] can be directly transferred from the implicit networks used for calculating the measures to the assessed community structure. This indicates that the assessed quality of the implicit network is an indirect indicator for the quality of the present community structure.



(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

4 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. The

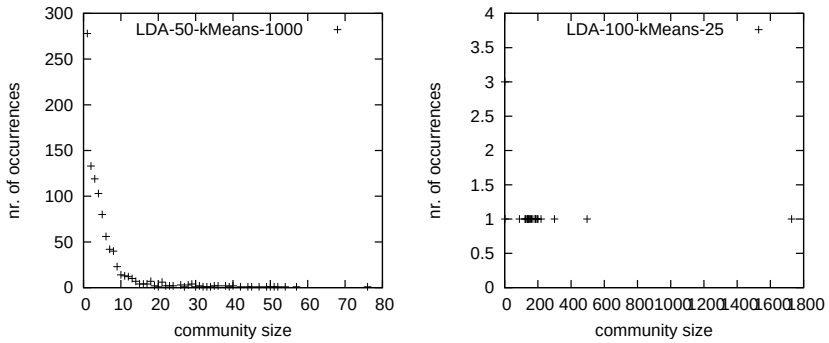


Fig. 6. Two opposed community size distributions as preferred by conductance (left) and modularity (right).

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 3. 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
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

Table 4. 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.

evaluation of this 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.

For future work, we aim to investigate, how the single evidence networks can be suitably combined into a weighted network. For this, we need to further analyze the individual structure of the networks, and the possible interactions. Furthermore, we plan to extend our experiments for a larger count of networks and clusterings in order to generalize the obtained results further.

References

1. Almendral, J.A., Oliveira, J., López, L., Mendes, J., Sanjuán, M.A.: The Network of Scientific Collaborations within the European Framework Programme. Physica A: Statistical Mechanics and its Applications 384(2), 675 – 683 (2007)
2. Baeza-Yates, R., Tiberi, A.: Extracting Semantic Relations from Query Logs. In: Proc. 13th ACM SIGKDD Conference. p. 85. ACM (2007)
3. Benz, D., Hotho, A., Jäschke, R., Krause, B., Mitzlaff, F., Schmitz, C., Stumme, G.: The Social Bookmark and Publication Management System BibSonomy – A Platform for Evaluating and Demonstrating Web 2.0 Research. VLDB. In Press. (2010)
4. Beyer, K.S., Goldstein, J., Ramakrishnan, R., Shaft, U.: When Is "Nearest Neighbor" Meaningful? In: ICDT. LNCS, vol. 1540, pp. 217–235. Springer (1999)
5. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet Allocation. JMLR 3, 993–1022 (2003)
6. 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)
7. Cattuto, C., Schmitz, C., Baldassarri, A., Servedio, V., Loreto, V., Hotho, A., Grahl, M., Stumme, G.: Network Properties of Folksonomies. AI Comm. 20(4), 245–262 (2007)
8. Clauset, A., Shalizi, C.R., Newman, M.E.J.: Power-Law Distributions in Empirical Data. SIAM Review 51(4) (2009)
9. Diestel, R.: Graph Theory. Springer, Berlin (2006)
10. Gaertler, M.: Clustering. In: Brandes, U., Erlebach, T. (eds.) Network Analysis. Lecture Notes in Computer Science, vol. 3418, pp. 178–215. Springer (2004)
11. 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)
12. 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 (2008)
13. Leicht, E.A., Newman, M.E.J.: Community Structure in Directed Networks. Phys. Rev. Lett. 100(11), 118703 (Mar 2008)
14. Leskovec, J., Lang, K.J., Mahoney, M.W.: Empirical Comparison of Algorithms for Network Community Detection (2010), cite arxiv:1004.3539
15. 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 (1967)
16. Mislove, A., Marcon, M., Gummadi, K., Druschel, P., Bhattacharjee, B.: Measurement and Analysis of Online Social Networks. In: 7th ACM SIGCOMM. p. 42. ACM (2007)
17. 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: Proceedings of the 21st ACM conference on Hypertext and Hypermedia. Toronto, Canada (2010)
18. Newman, M.E., Girvan, M.: Finding and Evaluating Community Structure in Networks. Phys Rev E Stat Nonlin Soft Matter Phys 69(2), 026113.1–15 (2004)
19. Newman, M.E.J.: Detecting Community Structure in Networks. Europ Physical J 38 (2004)
20. Newman, M., Park, J.: Why Social Networks are different from Other Types of Networks. Physical Review E 68(3), 36122 (2003)
21. 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, NY, USA (2010)
22. Siersdorfer, S., Sizov, S.: Social Recommender Systems for Web 2.0 Folksonomies. In: Proc. 20th ACM Conf. on Hypertext and Hypermedia. pp. 261–270. ACM, NY, NY, USA (2009)
23. Szomszor, M., Cattuto, C., Van den Broeck, W., Barrat, A., Alani, H.: Semantics, Sensors and the Social Web: The Live Social Semantics Experiments. Proc. ESWC 2010. pp. 196–210

Exploring country level gender differences in the context of online dating using classification trees

Slava Kisilevich and Mark Last

Department of Computer and Information Science
Konstanz University
slaks@dbvis.inf.uni-konstanz.de***
Department of Information System Engineering
Ben-Gurion University of the Negev
mlast@bgu.ac.il†

Abstract. The key component of Social Networking Sites (SNS), gaining increasing popularity among Internet users, is the user profile, which plays a role of a self-advertisement in the aggregated form. While computer scientists investigate privacy implications of information disclosure, social scientists test or generate social or behavioral hypotheses based on the information provided by users in their profiles. Statistical analysis of the SNS phenomenon often is performed using only a very small sample of information extracted from a particular SNS or by interviewing students from a particular university. In this paper, we apply classification algorithm to a large-scale SNS dataset obtained from more than 10 million public profiles with 50 different attributes extracted from one of the largest dating sites in the Russian segment of the Internet. In particular, we build gender classification models for the residents of the most active countries, and investigate the particular differences between genders in one country and the differences between the same-genders in different countries. The preliminary results are reported in this paper. To the best of our knowledge, this is the first attempt to conduct a large-scale analysis of SNS profiles and compare gender differences on a country level.

Key words: Social Networking Sites, Self-disclosure, Gender differences, Classification trees

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

*** http://www.informatik.uni-konstanz.de/arbeitsgruppen/infovis/mitglieder/slava-kisilevich/

† http://www.bgu.ac.il/~mlast/

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 SNS 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 SNS allow the user to upload at least one photo. Most mainstream SNS 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 user profiles or to conduct a survey among students (usually less than 100) of a particular university by presenting descriptive statistics of the sample data and performing significance tests between dependent variables [2, 10, 3, 8]. 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 million users, the results of the statistical analysis cannot be generalized for the whole population and theories can hardly be validated using only small samples. Moreover, Social Networks are heterogeneous systems, with people living in different parts of the world. To the best of our knowledge, the state of the art Social Science research of Social Networks does not take into account the spatial characteristics of the population. For example, due to cultural differences, the theory of self-disclosure tested on students from American universities may be not valid if applied on information obtained from students of Chinese universities, even if both groups use the same Social Network. 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 do 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 50 different attributes extracted from one of the largest dating site 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 between same-genders in different countries. 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 the minimizing of risks of online dating for the viewer of the profile. For this reason, in the broad context, assuming that the goal of the member of the 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 SNS using information extracted from user profiles that usually contain tens of different categorical and numerical attributes.

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

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 [13, 14], online relationships [5], ethnic identity [8], blogging [10], self-disclosure and privacy [2–4]. Since we could not find any related work on large-scale analysis of gender differences in social networks, we are going to review some of the recent studies and findings about gender differences in general.

Information revelation, privacy issues and demographic differences between users of Facebook SNS 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 reveal in their profiles. It was found that there is no difference between males and females with respect to their privacy attitudes and the likelihood of providing certain information. Likewise, there is no difference between genders in information revelation. If some information is provided, it is likely to be complete and accurate. However, female students are 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 difference 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 are 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 disclose 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 receive 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 have more friends and are more likely interested in friendship than males, but males are 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 send and receive 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. [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 are very important for establishing connections. These cues include 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 provide accurate information about themselves in the profiles.

3 Data

The data used in this paper was collected from one of the biggest 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 with the specially paid for VIP-status may get messages only from other VIP users. The user may exclude his/her profile to be searchable, but most of the profiles are searchable and accessible to unregistered users.

The user profile consists of seven sections also called blocks, where every block 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 "no" in the Single selection column), other attributes contain multiple selections (denotes as "yes" in the Single selection column). Most of the attributes also contain an additional open field that allows the user to provide his/her own answer. The user 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.* In addition, the user can provide additional information about himself/herself to assure that he/she is a real person. For this, he/she should send a free SMS to the company and confirm his/her mobile number.

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 including fields of the internal survey.

In 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.

[1] http://www.mamba.ru/
[2] http://www.monamour.ru/

Table 1. Profile sections and attributes

Section	Attributes	# of options	Single selection	Example
Personal	Age	-	yes	20
	Gender	2	yes	Male
	Zodiac	12	yes	Capricorn
Acquaintances	Seek for	5	no	Seek for a man of age 16-20
	Aim	13	no	Friendship and chatting
	Marriage	5	yes	Married and live together
	Material support	4	yes	I am ready to become a sponsor
	Kids	5	yes	I have kids, we live together
Type	Weight	1	yes	70 kg.
	Height	1	yes	180 cm.
	Figure	8	yes	Skinny
	Body has	2	no	Tattoo, Piercing
	Hair on the head	8	yes	Light colored
	Hair on the face or body	8	no	Chest, Hands
	Profession	-	-	Open field
	Day regimen	3	yes	I get up early
	Languages	87	no	English, German
	Economic conditions	5	yes	Wealthy
	Dwelling	7	yes	I live with my parents
	Life priorities	8	no	Carrier, Wealth, Family
Interests	Leisure	14	no	Reading, Sport, Party
	Interests	19	no	Science, Cars, Business
	Sports	12	no	Fitness, Diving
	Music	11	no	Rock, Rap
	Religion	7	no	Christianity, Atheism
	Smoking	5	no	I rarely smoke
	Alcohol	4	no	I like to drink
	Drugs	9	no	I never tried
Car	Car	76	yes	Nissan
Mobile	Mobile	50	yes	Ericsson
Sexual preferences	Orientation	4	yes	Hetero, Bi
	Heterosexual experience	6	yes	Yes, we lived together
	Frequency	7	yes	At least once a day
	Excitement	14	no	Smells, latex, tattoos

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 good representation 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 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 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 16 to 50. Due to the large number of profiles in Russia, we extracted no more than $20,000$ profiles for every age value and gender on every filtering step.

4.2 Data transformation

Almost all the attributes described in Table 1 were selected for inclusion into the model (except for *Weight* and *Height*). Numerical attributes include age, number of photos and number of words, whose length is more than two, used in the "About me" section. Attributes such as *Figure, Music, Car* or *Body has* whose values are not important for classification but only the fact of their presence or absence, were encoded as binary attributes: if the person provided information about his figure, it was coded as binary *True*, otherwise it was treated as *False*. On the other hand, attributes, whose values are relevant for classification were encoded as multi-valued categorical attributes. For example, the *Marriage* attribute has four explicit options (*I am married, we live together; I am married, we do not live together; I have a fictional marriage; No, I am not married*) and one implicit *no answer*. In this case the four options were coded like *1,2,3,4*, while in the case of implicit answer it was treated as a missing value. 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 answer categories. For example, the user can provide 13 different answers related to his aim on the site (*Aim* attribute). These 13 answers are categorized into six categories: *Friendship, Love, Sex, Sex for Money, Marriage* and *Other*. In this case, if the person provided his answer on the question from the

Table 2. Demographic statistics of the 20 most active countries and statistics related to the sampled data

Country	Total	Males %	Females %	# instances	Sampled Males %	Sampled Females %
Russia	7,999,976	35	65	1,332,563	40	60
Ukraine	1,294,260	48	52	813,322	17	83
Kazakhstan	473,561	43	57	222,579	18	82
Belarus	328,029	55	45	264,131	20	80
Germany	129,732	57	43	71,586	16	84
Azerbaijan	107,125	81	19	20,183	35	65
Uzbekistan	89,709	78	22	26,788	27	73
Moldova	84,306	59	41	54,561	15	85
Armenia	70,362	58	42	12,308	41	59
Georgia	69,805	80	20	18,163	26	74
Latvia	54,521	41	59	33,310	14	86
Estonia	49,030	48	52	27,991	16	84
USA	47,741	60	40	25,702	15	85
Israel	43,001	63	37	23,481	26	74
England	36,261	39	61	12,525	20	80
Lithuania	35,270	41	59	17,243	14	86
Turkey	35,230	84	16	9,003	26	74
Kyrgyzstan	35,107	64	36	16,263	21	79
Italy	18,681	58	42	12,495	10	90
Spain	18,619	61	39	9,919	12	88

Friendship category, a binary *True* is assigned to that attribute, otherwise *False* is assigned. Two binary attributes that were composed from the *Seek for*, namely *Seek for a man* and *Seek 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 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 determine 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 differences impact the characteristics of user profiles even for people of the same gender. The data mining process that can capture unique characteristics of the genders is a decision tree learning, which is based on model construction using input variables and prediction of the target class value (gender in our case).

We applied C4.5, a popular decision tree induction algorithm on the sampled data for every country with the *gender* as a binary class attribute, using Weka

data mining package [15]. We set the minimum number of instances per leaf to 10 and left all other options in their default state (pruned decision tree, 0.25 pruning confidence factor). Table 3 shows, for every country, the total number of rules generated by the algorithm, the number of rules per gender, the number of frequent rules (the rules that classify more than 100 instances) and the number of rules that cover more than 90% of the sampled data.

Table 3. The total number of rules generated by country, gender, the number of rules that classify more than 100 individuals (Frequent Rules), the number of rules that cover 90% of the instances in the sampled dataset

Country	**Rules**	**Male**	**Female**	**Frequent Rules Male**	**Frequent Rules Female**	**90% Rule Coverage Male**	**90% Rule Coverage Female**
Russia	7,462	3,747	3,715	619	688	1135	732
Ukraine	2,957	1,458	1,499	151	313	666	116
Kazakhstan	1,075	513	562	47	113	251	68
Belarus	1,372	654	718	64	143	340	92
Germany	429	200	229	14	49	128	39
Azerbaijan	221	101	120	14	14	50	38
Uzbekistan	191	96	95	15	20	47	19
Moldova	250	119	131	12	25	68	15
Armenia	147	75	72	6	8	39	25
Georgia	151	74	77	9	11	42	13
Latvia	177	79	98	4	23	55	17
Estonia	205	48	91	3	28	62	26
USA	175	77	98	5	23	52	20
Israel	242	114	128	11	23	64	34
England	95	39	56	4	15	27	16
Lithuania	106	47	59	2	12	34	11
Turkey	91	46	45	3	5	30	11
Kyrgyzstan	104	51	53	8	11	29	10
Italy	70	41	29	1	11	20	8
Spain	67	27	40	0	8	20	5

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 hypothesis that there are differences between genders and that these differences are also country-dependent.
The analytical steps are:
(1) Observation of the sampled data
(2) Observation of the quantity of rules that classify females and males

(3) Gender comparison
(4) Classification rules matching
(5) Gender characterization

5.1 Data observation

As was mentioned in Section 2, we applied four filtering steps to minimize the effect of false profiles. By inspecting the resulting number of females and males (Table 2), we can see the genders differences with respect to the profile creation. Many more females than males use different means of describing themselves through additional surveys, and many more females than males upload their photos. The largest difference between females and males can be observed in such countries as Italy (80%), Spain (76%), Latvia and Lithuania (72%), Moldova and USA (70%), while the smallest difference is in Armenia (18%), Russia (20%) and Azerbaijan (30%).

5.2 Model observation

The inspection of the quantity of generated rules that classify females and males (Table 3), shows that rules that classify females outnumber rules that classify males in 15 cases (countries), with the largest difference in Belarus. This finding may suggest that female users are more creative in profile construction and provide more heterogeneous information about themselves, while males use more homogeneous information to describe themselves. Moreover, the number of frequent rules is higher for females (19 cases) with the largest difference in Ukraine. This may also suggest the female users in different countries have more homogeneous behavior than men since they can be classified by relatively large amount of frequent rules. On the other hand, male users are heterogeneous with respect to the information they provide in their profiles, since most of them are classified by infrequent rules. This hypothesis is supported by inspecting how many rules cover the majority of the population. In all the cases, the number of rules that cover 90% of the population is larger for males with the greatest difference in Ukraine.

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. Inspection of the root attributes of the models reveals four groups of countries:
(1) Russia, Italy and Israel are characterized by the attribute *AimSex* (the aim on the site is to find a partner for having sex).
(2) Personality test (MonAmour test with more than 100 questions) is important for people from Azerbaijan and USA. This may suggest that people from Azerbaijan and USA consider the online dating as a very serious opportunity to find a romantic partner.
(3) Turkey is the only country where the classification tree is splitted according to the *Car* attribute. This may be explained by two reasons: (1) the number of

males is very high compared to the number of women and (2) most of the men like to "show off" by specifying what type of car they have.
(4) All other countries are characterized by the *photo* attribute that specifies how many photos were uploaded by a person.

5.3 Gender comparison

In Section 4.3 we applied a decision tree construction process to the user profiles from every country, 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 classifier's predictive performance, the model is evaluated on a separate test set. In the context of our analysis, we have evaluated the applicability of classification rules generated for each country to the data of other 19 countries. The high classification rate in this case should suggest that there is a high similarity between user profiles (and consequently between genders) across countries. We used 10-fold cross validation to estimate the testing accuracy of each country model on the data from the same country and used this result to report classification accuracy of the country's model. We selected 10% of profiles from the dataset of Russia and Ukraine using *StratifiedRemoveFolds* filter because Weka failed to run 10-fold cross validation on the entire dataset. Table 4 shows the classification accuracy for every model arranged in rows. The numbers on the diagonal represent the testing accuracy of each country model. The numbers arranged in columns represent classification accuracy of each of the country models on a test set of a given country. For simplicity of inspection, cells that have classification accuracy higher than 90% are colored in dark yellow, while cells that have classification accuracy less than 80% are colored in pink. It should be noted that the results are not symmetric. For example the classification accuracy of the Russian model on Moldova profiles is 79.93%, while it is 75.55% when Russian profiles are tested on the the Moldovan model. From the Table 4 we can see that the performance of most of the classifiers is around 80% to 90%. Classification accuracy of the Russian model on all other countries is below the average. Most classification models perform similar or even better on profiles from other countries than on profiles of their own countries, except for Azerbaijan and Armenia profiles, which have a lower performance with models of many other countries. Most accuracy differences in Table 4 were found statistically significant at the 99.9% confidence level.

The classification accuracy allows us to reason about cross-country prediction performance of each model, but it is not sufficient for comparing the countries behavioral patterns due to the non-symmetrical matrix (Table 4). The answer to the question "which countries are similar" is obtained by using the weighted Kappa statistic[3] [16–18], which is a measure of agreement between any two classifiers and defined as

[3] We calculated Kappa statistic using MedCalc statistical package `http://www.medcalc.be/`

$$K = \frac{P(A) - P(E)}{1 - P(E)} \tag{1}$$

where P(A) is the proportion of times that the classifiers agree and P(E) is the proportional agreement expected by chance.

Table 5 shows the Kappa values between countries. The interpretation of Kappa values was adopted from [19]. Cells that denote a *very good agreement* (0.801-1.0) are colored in light red, *good agreement* (0.601-0.8) are colored in blue, *moderate agreement* (0.401-0.6) are colored in green and cells that represent *fair agreement* are colored in yellow. According to [19] values below 0.20 represent poor agreements. Accordingly, we assume that there is no agreement between classifiers when Kappa values are below 0.20.

We can see that Belarus and Ukraine are the only countries with a *very good* agreement (0.826), which indicates that behavioral patterns are very similar in these countries. Germany has the largest number of *good* agreements (four in total). Armenia and Lithuania are the countries that have the largest number of *moderate* agreements (six in total). Kazakhstan and Italy are the countries that have the largest number of *fair* agreements (seven in total). Kazakhstan, Belarus, Germany, Armenia, Latvia and Lithuania are the country with the most number of agreements (eleven in total). Russia, on the other hand, is the only country, which does not have any similarities to other 19 countries.

5.4 Rule matching

The gender classification rules that were generated for every model (Table 3) consist of the most important attributes and values that characterize females and males in a specific country. If we expect to have a similarity between same genders in different countries, then there should be a high number of similar rules found in the models. We performed the comparison of classification rules (rule matching) by taking every rule in a model, and searching for rules that have the same attributes and values in the precedent of the rule in any order in other models. For example the following two rules $A{=}x$ *AND* $B{=}y \rightarrow c$ and $B{=}y$ *AND* $A{=}x \rightarrow b$ match because they have common attributes (A and B) in the precedent and those attributes take the same values.

Due to the space limitation we cannot provide the complete results of the comparison. However, it should be noted that the number of matching rules is very low. For example, the highest number of matching rules was observed between Latvia and England, and Germany and Israel (4 rules).

Latvia-England:

(1) *If there are no photos AND aim is sex → males*

and 3 rules that classify females:

(2) *If there is at least 1 photo AND aim is not sex AND have a car AND seek for a person older than 21 AND no kids AND no information about the body → females*

(3) *If there is at least 1 photo AND aim is not sex AND have a car AND seek for a person older than 21 AND have kids living together → females*
(4) *If there is at least 1 photo AND aim is not sex AND have a car AND seek for a person older than 36 AND have kids but live separately → females*

We can clearly see the differences between females and males on the example of the rules presented above. While males are characterized by intimate intention (to have sex) and lack of photos, females are characterized by availability of at least one photo in their profiles and the information regarding the desired age of the partner. It is possible that young female users who do not have children or those who have children search for a person older than 21, while older female users who have children not living in the same household, would like to meet a person older than 36.

Israel and Germany are another two countries that have four common rules. One of the precedents of the rule is the following:
If there is at least 1 photo AND aim is sex AND personality test is filled AND sexual orientation is Bisexual AND no kids
The German model classifies this rule as *females*, but the Israeli rule classifies this rule as *males*. It should be noted that except for this ambiguous classification due to specificity of sexual orientation, all other common rules are not ambiguous. This is a good indication that there is a consistency in common rules among the same genders across different countries.

5.5 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.
Azerbaijan: *If personality test is filled AND aim is sex → males*
Azerbaijan: *If personality test is not filled AND there is at least 1 photo AND have a car AND younger than 25 AND seek for a person older than 18 AND the body is slim → females*
Belarus: *If no photos AND aim is sex AND seek for a person younger than 22 → males*
Belarus: *If there is at least 1 photo AND aim is not sex AND no car AND personality test is not filled AND preferences in sex are provided AND seek for a person older than 21 AND have kids living together → females*
Russia: *If aim is sex AND seek for a person younger than 22 AND sexual orientation is Heterosexual AND older than 18 AND younger than 42 AND personality test is filled → males*
Russia: *If aim is not sex AND there is at least 1 photo AND seek for a person older than 25 AND older than 29 AND do not smoke AND have kids living together → females*

From these examples we can see that the sex component is present in the male rules (Azerbaijan, Belarus, Russia), photos are uploaded more by females (Belarus, Russia). In addition, the difference in age of the female and the person

she seeks for is not significant (Azerbaijan, Russia), while males specify young females as their desired romantic partners (Belarus, Russia)

6 Conclusions

In this paper we investigated gender differences between countries in the context of dating sites using approaches from the field of Data Mining. We applied decision tree construction algorithm to the user profiles from 20 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 generated models and found countries where users behave similarly in terms of profile creation. However, the majority of countries are different from each other, which suggests that cultural aspects influences the way people behave in social networking sites. We also analyzed the induced classification rules and found almost no similarity between the same genders from different countries. This fact reinforces our hypothesis that cultural aspects influences behavior not only of different genders across countries but also of people of the same gender. We showed that social phenomena can be investigated using data mining methods if large quantities of data are available, and when 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.

The preliminary results provided in the paper are encouraging. In our future work, we will apply more analytical methods to conduct all-embracing gender difference analysis and work closely with social scientists to test hypotheses that so far have been verified on very limited amounts of sampled data.

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) (2009) 2
2. Acquisti, A., Gross, R.: Imagined communities: Awareness, information sharing, and privacy on the Facebook. In: Privacy Enhancing Technologies, Springer (2006) 36–58

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, ACM (2009) 265–274
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 (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) (2006) 415
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) (2008) 1321–1330
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) (2009) 643–654
8. Grasmuck, S., Martin, J., Zhao, S.: Ethno-Racial Identity Displays on Facebook. Journal of Computer-Mediated Communication **15**(1) (2009) 158–188
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) (2007) 1472
11. Goodchild, M., Anselin, L., Appelbaum, R., Harthorn, B.: Toward spatially integrated social science. International Regional Science Review **23**(2) (2000) 139
12. Kleinberg, J.: The convergence of social and technological networks. Commun. ACM **51**(11) (2008) 66–72
13. Golub, Y., Baillie, M., Brown, M.: Gender Differences in Internt Use and Online Relationships. American Journal of Psychological Research **3**(1) (2007)
14. 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) (2009) 244–264
15. 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) (2009) 10–18
16. Cohen, J.: A coefficient of agreement for nominal scales. Educational and psychological measurement **20**(1) (1960) 37
17. Cohen, J.: Weighed kappa: Nominal scale agreement with provision for scaled disagreement or partial credit. Psychological Bulletin **70** (1968) 213–220
18. Fleiss, J., Levin, B., Paik, M.: Statistical methods for rates and proportions. NY John Wiley & Sons (2003)
19. Altman, D.: Practical statistics for medical research. Chapman & Hall/CRC (1991)

	Russia	Ukraine	Kazakhstan	Belarus	Germany	Azerbaijan	Uzbekistan	Moldova	Armenia	Georgia	Latvia	Estonia	USA	Israel	England	Lithuania	Turkey	Kyrgyzstan	Italy	Spain
Russia	83.08	79.12	77.54	79.04	75.02	72.38	75.31	75.55	71.88	73.04	73.34	74.25	73.02	77.04	73.41	72.50	72.81	74.80	70.53	70.36
Ukraine	85.85	89.10	88.53	88.98	87.19	78.56	85.00	87.30	79.14	82.31	86.91	87.28	85.90	85.90	86.60	86.15	85.01	86.39	84.70	85.68
Kazakhstan	86.16	88.37	89.75	89.93	86.11	78.53	84.42	86.25	78.22	81.86	85.39	85.85	84.60	85.05	85.29	84.79	82.72	85.43	83.64	84.40
Belarus	86.37	89.84	90.73	88.89	87.28	80.97	85.90	87.53	80.63	83.83	86.94	87.13	85.89	85.90	86.71	86.37	84.94	87.06	84.88	85.68
Germany	85.66	89.63	88.43	88.78	88.94	79.64	86.17	87.27	80.73	84.49	87.49	87.86	87.30	86.53	87.08	86.95	83.39	86.44	86.25	86.25
Azerbaijan	88.25	91.07	90.36	90.94	89.34	83.93	88.57	91.62	82.68	86.19	89.12	89.46	88.50	88.71	88.60	88.85	86.70	89.19	87.75	88.42
Uzbekistan	85.51	86.38	84.11	85.49	83.85	76.24	88.59	83.06	77.54	80.42	82.09	83.57	82.51	88.21	82.47	81.06	80.88	82.17	80.46	79.39
Moldova	79.93	80.42	80.23	81.16	76.73	86.62	82.18	90.65	81.80	80.01	75.15	76.21	73.14	78.11	76.34	75.29	78.20	79.38	72.66	74.91
Armenia	86.72	89.73	88.45	89.40	88.35	77.16	84.27	87.52	83.20	82.30	87.85	90.39	87.20	87.11	87.32	87.59	84.25	86.04	85.82	86.50
Georgia	85.42	86.96	86.57	87.41	84.19	84.78	90.11	86.87	84.60	86.17	83.31	83.88	83.36	84.48	84.80	83.93	83.36	86.54	81.42	83.18
Latvia	86.95	90.83	89.43	89.89	89.37	78.36	86.18	88.30	79.45	84.06	89.47	89.32	88.78	88.15	88.57	88.34	85.09	87.51	87.55	87.91
Estonia	84.50	88.63	87.17	87.43	87.56	78.68	85.52	85.64	79.32	83.26	86.72	88.64	91.13	85.13	85.90	85.92	82.41	85.94	86.81	85.39
USA	80.76	85.89	85.48	85.37	83.82	83.97	85.91	84.89	84.12	88.04	82.94	83.09	89.63	83.07	83.34	83.67	82.44	84.79	80.36	82.67
Israel	79.01	80.67	80.87	81.42	77.15	82.26	81.82	79.36	85.55	80.83	76.04	77.05	71.22	85.82	77.45	76.79	78.18	79.64	71.07	75.83
England	86.98	89.55	89.09	90.00	87.30	83.71	87.55	88.53	83.40	85.62	87.12	87.23	85.82	86.69	86.43	86.68	84.87	90.10	84.28	86.10
Lithuania	85.26	90.48	89.49	89.80	89.29	78.25	85.99	88.34	79.29	84.35	88.75	88.98	88.48	86.79	88.13	88.84	83.81	87.44	87.10	87.86
Turkey	83.95	87.76	86.47	87.12	85.68	78.06	84.63	85.33	79.32	82.24	84.89	85.21	85.21	84.87	87.87	84.22	85.69	85.06	83.87	83.65
Kyrgyzstan	84.81	85.60	85.29	85.84	83.18	82.23	83.84	83.09	82.41	81.96	81.31	82.38	81.86	84.01	82.66	80.52	88.16	88.72	80.68	79.05
Italy	84.82	91.11	89.98	89.80	89.88	78.89	87.13	89.44	81.81	85.22	89.39	89.64	89.60	88.70	89.72	89.23	85.50	88.82	91.80	91.29
Spain	86.30	92.32	91.75	91.88	91.23	80.54	89.52	91.35	83.81	87.18	91.62	91.49	91.82	89.74	91.12	91.14	87.24	90.60	93.05	89.87

Table 4. Classification accuracy of 20 most active countries.

Russia	Ukraine	Kazakhstan	Belarus	Germany	Azerbaijan	Uzbekistan	Moldova	Armenia	Georgia	Latvia	Estonia	USA	Israel	England	Lithuania	Turkey	Kyrgyzstan	Italy	Spain
0.003																			
0.006	0.514																		
0.001	0.826	0.464																	
0.001	0.634	0.369	0.606																
0	0.189	0.182	0.271	0.277															
0.027	0.118	0.254	0.089	0.086	0.032														
0.173	-0.064	-0.071	-0.024	-0.059	0.032	-0.96													
0.003	0.598	0.44	0.555	0.715	0.32	0.06	-0.054												
0	0.095	0.278	0.085	0.06	0.014	0.229	0.047	-0.05											
0.002	0.361	0.32	0.419	0.481	0.565	0.092	-0.071	0.547	-0.057										
0.003	0.528	0.39	0.469	0.535	0.133	0.108	-0.112	0.468	0.035	0.359									
-0.02	-0.127	-0.024	-0.102	-0.033	-0.026	0.144	0.032	-0.113	0.359	-0.042	0.045								
0.129	-0.081	-0.118	-0.055	-0.06	-0.008	-0.077	0.658	-0.044	-0.002	-0.072	-0.11	0.003							
0	0.601	0.358	0.677	0.485	0.347	0.064	0.004	0.436	0.172	0.368	0.264	-0.046	-0.014						
0.002	0.455	0.351	0.488	0.606	0.561	0.083	-0.072	0.605	-0.017	0.787	0.422	-0.021	-0.078	0.415					
0.003	0.388	0.528	0.321	0.216	0.093	0.17	-0.099	0.261	0.123	0.176	0.328	0.015	-0.129	0.165	0.183				
0.015	0.051	0.118	0.033	-0.037	0.036	0.421	0.015	0.003	0.201	-0.02	-0.002	0.091	0.059	0.031	-0.019	0.219			
0.002	0.252	0.192	0.327	0.325	0.483	0.073	-0.047	0.334	-0.013	0.484	0.286	-0.007	-0.058	0.253	0.489	0.164	0.036		
0	0.139	0.112	0.155	0.177	0.332	0.052	-0.014	0.195	0.007	0.27	0.144	-0.006	-0.026	0.175	0.232	0.101	0.034	0.352	

Table 5. Agreements between classifiers using Kappa statistics. Light red: very good agreement (0.801 to 1.0), Blue: good agreement (0.601 to 0.8), Green: moderate agreement (0.401 to 0.6), Yellow: fair agreement (0.201-0.40)

Mining Social Context with Wearable Sensors

Ciro Cattuto

Complex Networks Lagrange Laboratory,
Institute for Scientific Interchange (ISI) Foundation,
10133 Torino, Italy
ciro.cattuto@isi.it

Abstract. The availability of networked wearable devices is providing new ways to expose the mobility and interaction patterns of individuals. We describe a scalable RFID-based platform that was designed to sense the proximity relations of individuals, providing a real-time representation of their social context. We report on the experience of deploying this platform at conferences, museums, hospitals and company offices, and show how proximity information can be used for a variety of applications. In particular, we describe an application that supports and augments social interactions at conference gatherings by mining and linking real-world physical proximity with information from online social networking systems.

PREDICTABILITY OF MOBILE PHONE ASSOCIATIONS

Bjørn Sand Jensen, Jan Larsen, Kristian Jensen,
Jakob Eg Larsen, and Lars Kai Hansen

Technical University of Denmark
{bjje,jl,krije,jel,lkh}@imm.dtu.dk

Abstract. Prediction and understanding of human behavior is of high importance in many modern applications and research areas ranging from context-aware services, wireless resource allocation to social sciences. In this study we collect a novel dataset using standard mobile phones and analyze how the predictability of mobile sensors, acting as proxies for humans, change with time scale and sensor type such as GSM and WLAN. Applying recent information theoretic methods, it is demonstrated that an upper bound on predictability is relatively high for all sensors given the complete history (typically above 90%). The relation between time scale and the predictability bound is examined for GSM and WLAN sensors, and both are found to have predictable and non-trivial behavior even on quite short time scales. The analysis provides valuable insight into aspects such as time scale and spatial quantization, state representation, and general behavior. This is of vital interest in the development of context-aware services which rely on forecasting based on mobile phone sensors.

1 Introduction

The wide acceptance of sensor rich mobile phones and related applications enables deep studies of human behavior. According to a recent study by Song et al. [11, 12] based on mobile phone location trajectories, individual human mobility patterns are highly predictable. When including the complete history of the participants they derived an upper bound on prediction of the next location of 93% in a large cohort of 45,000 users. The upper bound is based on information theory. Using Fanos inequality it was shown in [12] that the entropy rate transforms into an upper bound of the predictability.

Interest in understanding human behavior using mobile technology is increasing, see e.g., the recent review by Kwok [9]. The work of Eagle et al. [4] on the Reality Mining dataset, marks an early and important contribution. Using a Hidden Markov model and the time of day, they demonstrate explicit prediction accuracies (home, work, elsewhere) in the order of 95%. Furthermore, they use principle component analysis (PCA) to visualize temporal patterns in daily life. The stability of these temporal patterns was confirmed by Farrahi et al. [5] in the same dataset using unsupervised topic models.

While Eagle et al. focus on finding statistical regularities in behaviors at the group level using parametric models Song et al. [11] are interested in individual predictability using non-parametric methods and argue that inter-participant variability is significant and in fact power-law distributed. For a further discussion on parametric and non-parametric models, and the relation to information theory, we refer to Bialek et al. [3].

We follow the implicit modeling approach by Song et al., i.e., using bounds rather than explicit predictors to discuss human behavior in a novel mobile phone data set that complements the analysis of Song et al. Our data set has significantly higher temporal resolution and involves more sensors, however, in a much smaller cohort ($N = 14$).

The opportunity to analyze multiple sensors is quite unique and produces new insight both on the predictability of each sensor and on sensor dependencies. We show that all the location and proximity based sensors have a relatively high predictability bounds for the entire population. Whereas Song et al. [11] provide important results on location prediction, the multiple sensors applied in our study can potentially provide a richer description of context beyond location. And as the extended set of sensors enjoys similar high predictability rates, it may contribute additional useful information on human behavior and support more general context dependent services.

Furthermore, we are interested in the predictability on different time scales, and to probe whether it is possible to predict non-trivial behaviors on smaller scales than the one hour time scale considered in [11]. Finally, we suggest applying mutual information - or prediction information [3] - as a method to easily estimate an optimal time scale in a non-parametric fashion. The information theoretic approach is based on upper and lower bounds on predictability. In this paper we use the upper bound proposed by Song et al., and derive and analyze a tighter lower bound based on a first Markov model [7], rather than the zero order model of [12].

An early version of this work was presented to the machine learning community in [7]. The present paper extends this work with; 1) an extended description of the experimental setup; 2) comparison between WLAN and GSM, in particular in relation to optimal time scale; 3) extended discussion and interpretation.

The paper first gives a presentation of the experimental setup and the acquired mobile phone dataset. In Section 3 we present the information theoretic tools necessary to follow the analysis of the dataset. The result of the study is presented and discussed in Section 4, followed by the conclusion in Section 5.

2 Experimental Setup

Within the last decade there has been a number of studies of real-world dataset or *lifelogs* reflecting human life [1, 2]. In this section we present a software platform for obtaining such *lifelogs* using standard off-the-shelf mobile phones functioning as individual wearable sensor platforms.

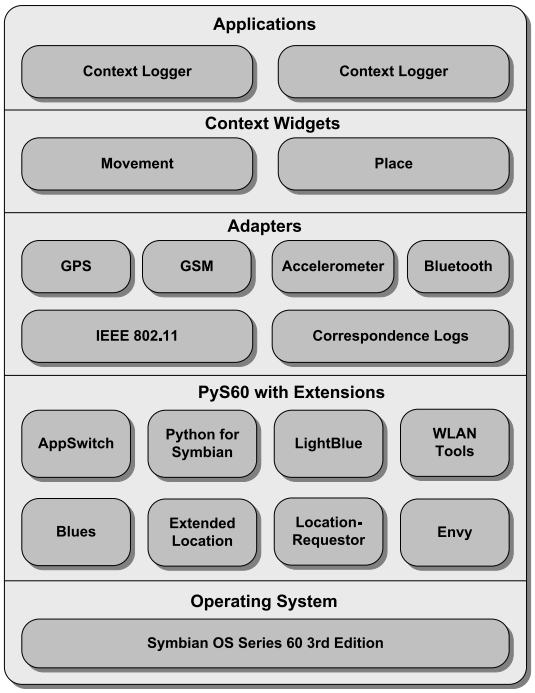


Fig. 1. Mobile Context Toolbox architecture [10]. The bottom two layers provide low-level access to the embedded sensors, whereas the adapters and context widget layers provide high-level Python interfaces to sensors and inferred information for applications.

2.1 Mobile Context Toolbox

Since utilizing multiple sensor inputs on mobile devices can be a complex task we have used our Mobile Context Toolbox [10], which provides an open extensible framework for the Nokia S60 platform running the Symbian mobile Operating System present in mobiles phones such as Nokia N95. The framework provides access to multiple embedded mobile phone sensors, including accelerometer, microphone, camera, etc., as well as networking components such as phone application data (calendar, address book, phone log, etc.), and phone state (profile, charge level, etc.).

In principle, mobile devices can acquire information from the surrounding environment as well as from online sources, but in the present study we focus on information that can be acquired through the large variety of sensors embedded in the device. The framework has multiple layers (as depicted in Fig. 1) on top of the Nokia S60 platform. The framework uses Python for S60 (PyS60) with a set of extensions for accessing low-level sensors and application data. The adapters layer provides interfaces for the low-level sensors, whereas the context widgets uses one or more adapters to infer higher-level contextual information. Finally,

the application layer utilize contextual information inferred from context widgets and/or directly from context adapters. Further details and explanation of the Mobile Context Toolbox is provided in [10].

2.2 Logging Data From Embedded Sensors

For the purpose of using mobile phones as an instrument for gathering *lifelog* information we have built a *Context Logger* application, which subscribes to all sensors through the relevant system components and then continuously records all events received from the multiple adapters and widgets. In effect all accessible sensor data is recorded, as shown in Table 1.

All sensor recordings are time stamped using the embedded mobile phone timer. The accelerometer was sampled every 2 seconds. Samples are read-out every 30s, in batches of 15 samples. The purpose of reading in bursts is to enable reading short bursts with higher sampling rate. GSM cellular information acquired included the Cell ID with country code, operator code, and location area code. In the present system the phone software API only allow reading the CellID of the GSM base transceiver station to which the phone is currently connected (not the ones visible). Since the GPS sensor is the most energy expensive sensor, it was only sampled 2–3 times an hour to reduce the energy consumption. Bluetooth scans was performed approximately every 1.5-3 minutes. The sampling rate varies as the Bluetooth discovery time increases with the number of Bluetooth devices available within discovery range. A discovery of a Bluetooth device will always produce the unique MAC address of the device, however, the lookup of the Bluetooth "friendly name" and device type might fail as more time is required to obtain this information. WLAN scanning is performed approximately once a minute, recording MAC address, SSID, and RX level (power ratio in decibels – a measurement of the link quality) of discovered Access Points. Finally, phone activity (SMS, MMS, and calls) were recorded whenever it occurred (phone number and direction).

In addition to the above mentioned sensor data, the *Context Logger* application allows a user to manually label his/her present location and activity. The label is a text string which can be entered by the user on the mobile phone, such as, *home*, *running*, and *having dinner*. Entered labels will be stored and subsequently shown on a list to pick from in order to avoid re-typing location and activity labels. Users can manually choose to label location and activity by

Sensor	Sampling	Data
Accelerometer	30/minute	3D Accelerometer values
GSM	1/minute	CellID of GSM base transceiver station
GPS	2–3/hour	Longitude, Latitude, and Altitude
Bluetooth	20–40/hour	Bluetooth MAC, friendly name, and device type
WLAN	1/minute	Access Point MAC address, SSID, and RX level
Phone activity	Event	Phone number and direction of call or message

Table 1. List of embedded mobile phone sensors used for collecting data

selecting a menu item in the *Context Logger* application, however, this mechanism is further enhanced with the ability to automatically prompt for labels. Thus, a user can choose to enter a label at any point in time, but the application will also prompt the user to label a location and activity 2–3 times a day in order to receive feedback. The data location and activity data recorded on the mobile phone is a key-value pair along with the time stamp. There are no pre-defined location and activity labels defined in the application and the labeling is completely free form with the users determining, how they want to write their text labels.

Situations with missing data may occur due to the phone running out of battery, being switched off, or simply not able to acquire data through one or more sensors (for instance no GSM reception).

2.3 Data Collection

An initial deployment of the system included continuous use by 14 participants, each equipped with a standard mobile phone (Nokia N95) which had the Mobile Context Toolbox pre-installed along with the *Context Logger* application that would continuously record the data acquired from all sensors currently supported by our framework. The participants were using the mobile phone as their regular mobile phone for a period of five weeks or more, as they were given instructions to insert their own simcard into the phone. Furthermore, they were instructed to make and receive calls, send messages, etc., as they would usually do, and generally use the phone as they would use their own phone. Therefore, no particular instructions were given, since we wanted to establish data from regular usage of the mobile phone, and thereby acquire real-life data. This means that the participants would not necessarily carry the phone on the body all the time such as carrying the mobile phone in a pocket.

As the survey is completely dependent on the cooperation of the participants and due to increased use of sensors, the lack of battery time was considered a risk in terms of participants leaving the survey. Thus, the sensor configuration of sampling on the phone was based on optimizing the resource consumption, so that the participants should only need to recharge the phone once a day (typically during the night).

The experiment started on October 28, 2008 and ended January 7, 2009 and the participants were students and staff members from The Technical University of Denmark volunteering to be part of the experiment. Thus, the participants were mainly situated in the greater Copenhagen area, Denmark. The 14 participants took part in the experiment between 31 to 71 days, resulting in approximately 472 days of data covering data collection periods totalling 676 days. The average duration was 48.2 days. An overview of the collected data is provided in Table 2.

During the experiment a total of approximately 20 million data points were collected with the accelerometer contributing the most with 14.5 million data points. A summary of recordings from Bluetooth scans, WLAN scans, GPS readings, and GSM readings can be seen in Table 2. It is worth noticing the

Part.	Accel	BT.	BT.*	GPS	GSM	GSM*	Ann.	PA.	WLAN	WLAN*	Days
1	1474480	54349	2846	516	69458	529	533	544	224101	6387	71
2	2045773	38028	2478	1514	75669	603	596	1062	364272	6040	66
3	318597	27329	790	12	37217	98	222	21	125600	630	31
4	875287	7880	743	2	17750	228	134	620	186421	2394	52
5	1117147	13575	2373	4058	56206	227	386	277	251016	2347	48
6	711490	23702	1141	95	51702	235	82	839	92396	2119	50
7	1184457	13327	1765	3	45826	955	272	581	139466	4017	46
8	700258	42346	2080	614	74250	172	212	74	154108	3359	41
9	1101926	42346	1050	119	37393	100	104	497	104576	1804	38
10	1103086	21676	2104	419	63937	419	414	116	192338	2650	48
11	1122315	12492	655	929	46158	929	163	121	295716	2286	47
12	796452	30610	2317	40	51548	40	143	151	97769	2403	50
13	1024276	27550	1741	1114	49349	1114	137	949	171951	5463	51
14	971558	21502	1303	44	40017	44	149	686	118687	1263	36
Total	14547102	350879	20408	9479	716480	2837	3547	6538	2518417	28110	48.2

Table 2. Overview of collected data for each participant in the experiment: Participant, Accelerometer, Bluetooth, Unique Bluetooth devices, GPS, GSM, Unique GSM cells, Annotations, Phone Activity, WLAN Access Points, Unique WLAN Access Points, Duration in days.

number of unique Bluetooth devices, unique GSM cells, and unique WLAN access points discovered accumulatively during the experiment by all participants: 20408, 2837, and 28110, respectively. In total 9479 readings of GPS position were recorded, but the recordings varies a lot among the participants due to the nature of the GPS technology. As a GPS position typically can not be obtained indoor only a subset of users have sufficient recordings of GPS position. For instance, if they typically place the mobile phone near a window when indoor where a GPS position can be obtained. A total of 6538 calls and messages took place during the experiment.

The participants provided 3547 annotations of locations and activities in total. On average the participants provided 253 annotations during their participation, with an overall average of 5.3 labels provided per user per day. The most active participants provided 8-9 labels per day on average, whereas the least active participants provided 2 labels per day on average.

3 Methods

In this study we will apply an information theoretic approach in the analysis of the dataset obtained and described in Section 2. Although before describing the details involved in this, we consider the preprocessing required for the final analysis.

A general issue in obtaining discrete times series is the number of quantization levels and sample rate of the true process [3]. The scan cycles used to obtain the

present dataset are non-uniformly sampled and the scan cycles are of different length for each sensor. We therefore construct a commonly aligned time series for each sensor by creating non-overlapping frames of a given window length. The original samples falling within the frame is then assigned to it. If multiple samples falls within a frame they are merged, which is reasonable for the indicator type of sensors (WLAN, GSM, BT). This is similar to combining states in a Markov model effectively altering the state transitions. In case of the acceleration sensor, the feature is calculated as the average power within a window represented as a discrete levels[1], i.e., $X^{ACC} \in \{\text{off}, 1, 2, 3\}$. Considering the WLAN sensor and integrating all the networks seen into a state effectively means that the predictive variable becomes the WLAN state and not the location as such. If a specific location is needed a lookup to a database could return the position of the WLAN access point and generate a location variable from that. An alternative would be to directly work on a location variable generated from the WLAN access point, but this is not considered here.

The proposed representation constitutes a very detailed description of the participant state. An alternative suggested in [11, 12], represents the state as the most visited GSM cell location within a time window. Both approaches involve a relatively complex temporal quantization and resampling of the original data. To evaluate the consequence temporal scale, we consider the change in predictability bounds as the window length is decreased from one hour to one minute.

3.1 Information Theoretic Measures

We consider the problem of quantifying the predictability obtainable in a discrete process, $\mathbf{X} = (X_1, X_2, .., X_i)$, where i is the time index and X is the state variable. This is to a large degree motivated by previous work on predictability, complexity and learning (see e.g. [3]), and recent development on a similar dataset [11]. Here predictability is defined as the probability of an arbitrary algorithm correctly predicting the next state. Hence, given the history the basic distribution of interest is $P(X_{i+1}|X_1, X_2, .., X_i)$. In the case where we have no information regarding the history, the distribution naturally reduces to $P(X)$. When $P(X)$ is uniform, i.e., each of M states have the same probability of occurring, the Shannon entropy (in bits) is defined as $H^{rand}(X) = \log_2 M$.

In the case where the distribution of X is non-uniform, the entropy is given as

$$H^{unc}(X) = -\sum\nolimits_{i \in I} p(x_i) \log(p(x_i)) \tag{1}$$

with $p(x) = P(X = x)$. This in turn represents the information when no history is available, hence, the acronym unc (uncorrelated).

The entropy rate of the participants trajectory, or the average number of bits needed to encode the states in the sequence, can be estimated taking into

[1] An equiprobable quantization is used, i.e., each level has the same frequency of occurrence within the entire dataset.

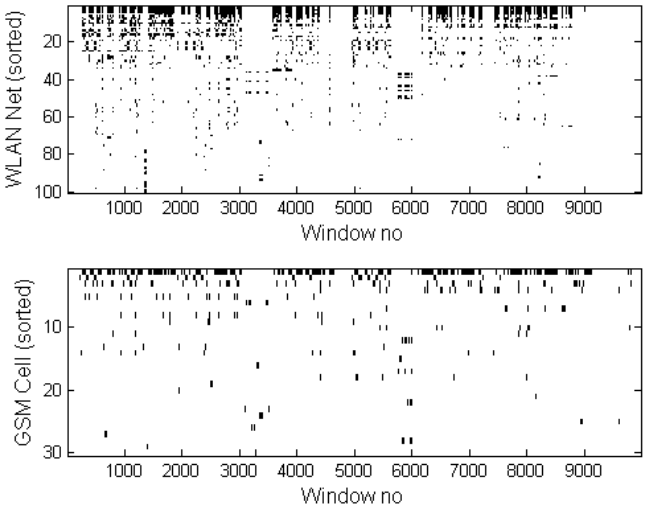


Fig. 2. Participant 2. Time series for WLAN (top 100) and GSM data (top 30). The time series are considered in a vector space representation, hence, each column is a state vector.

account the complete history. This is done by defining the stationary stochastic process $\mathbf{X} = \{X_i\}$ which have the entropy rate defined as

$$H(\mathbf{X}) = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} H\left(X_i | h^{i-1}\right), \tag{2}$$

where the history h^i at time step i is $h^i = \{X_1, X_2, ..., X_{i-1}\}$. It is noted that $0 \leq H \leq H^{unc} \leq H^{rand} < \infty$.

A challenge in using these information measures based on real and unknown processes is the estimation of the entropy rate. A number of ideas have emerged based on compression techniques such as Lempel-Ziv (LZ) (including string matching methods) and Context Weighted Trees for binary processes. For a general overview, we refer to [6]. An appealing aspect of these non-parametric methods is that we avoid directly limiting model complexity as would be necessary if we applied parametric or semi-parametric models. In this study we estimate the entropy rate using a LZ based estimator as described in [8, 6] and also applied in [12]. The entropy rate estimate for a time series of length n is given by

$$H^{est} = \left[\frac{1}{n} \sum_{i=1}^{n} \frac{L_i}{\log_2 n}\right]^{-1} \tag{3}$$

where L_i is the shortest substring starting at time step i, which has not been seen in the past. The consistency under stationary assumptions is proved in [8] where the method is applied to the analysis of English text.

Given estimates of the entropy and the entropy rate we consider a related quantity, namely mutual information - or predictive information [3]. This measure is already available given the information measures above as

$$I_{pred} = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} H\left(X_i\right) - H\left(X_i|h^{i-1}\right) \tag{4}$$

$$= H^{unc}\left(X\right) - H^{est}\left(\mathbf{X}\right) \tag{5}$$

In effect I_{pred} represents the mutual information between the distributions corresponding to knowing and not knowing the past history. Hence, it quantifies the fundamental information gain in knowing the (complete) past when prediction the future, and we propose it as a easy way to evaluate quantization and time scale effects. We illustrate the behavior of the measure on a time scale selection problem in Section 4.

3.2 Predictability

In order to construct bounds on the predictability we consider the probability, Π, that an arbitrary algorithm is able to predict the next state correctly.

Based on the entropy rate and Fano's inequality Song et al. derives a bound so $\Pi \leq \Pi^{\max}\left(H\left(X\right), M\right)$ with Π^{max} given by the relation [12]

$$H\left(X\right) = H\left(\Pi^{\max}\right) + \left(1 - \Pi^{\max}\right)\log_2\left(M-1\right) \tag{6}$$

with the function, $H\left(\Pi^{\max}\right)$, given by

$$H\left(\Pi^{\max}\right) = -\Pi^{\max}\log_2\left(\Pi^{\max}\right) - \left(1-\Pi^{\max}\right)\log_2\left(1-\Pi^{\max}\right) \tag{7}$$

This non-linear relation between Π^{max} and the estimate of $H(X)$ is easily solved using standard methods (for a full derivation see [12]).

Adopting this approach we obtain three upper bounds based on the entropy estimates previously mentioned. The first, Π^{rand}, provides an upper bound on a random predictor. The second upper bound is Π^{unc} which bounds the performance obtainable with a predictor utilizing the observed state distribution. Finally, the most interesting bound, Π^{max}, provides a upper limit for the performance of any algorithm utilizing the complete past.

The upper bound is of course interesting in understanding the potential predictability, although we find a lower bound equally important in the analysis. Song et. al. [11] show how a simple lower bound can be constructed based on the so-called regularity. The regularity is in essence a zero order Markov model based on the most likely state at any given time of the day, i.e., using only the time of occurrence and no sequence information. This is an intuitive measure for some time periods such as daily patterns, e.g., utilizing where a person is most likely to be each morning at 7.00. However, if the time scale is in the minute range it does not necessarily make sense to consider the regularity.

Instead we propose to use a predictor using the immediate past as the representative of the lower predictability bound. For this purpose we use a first order

Markov model with the transition probabilities estimated from the finite process. Thus, the next state prediction is based on the distribution $P\left(X_{i+1}|X_1, X_1, ..., X_i\right) = P\left(X_{i+1}|X_i\right)$.

To avoid overfitting which tends to render the bounds overly optimistic, we use a resampling scheme in which the entropy and the next state distribution is estimated based on 2/3 of the data and tested on the remaining 1/3. This is performed for nine distinct subsections in a compromise between accuracy of the estimate and the needed samples for the entropy estimator to converge. The resampling further allows us to verify that the LZ entropy estimate converges to reasonable similar values for separate temporal sections of the participants life. Any variation across subsections will result in a greater variance of the estimated bound.

4 Results

In order to provide insight into the predictability of mobile phones sensors and thereby indirectly insight into human behavior, we apply the information theoretic methods described in Section 2. The density estimates of the bounds are all made using a standard kernel based density estimator (the bandwidth is hand-tuned for visualization).

4.1 Individual Sensors

One of the goals in this study is to analyse the potential predictability of different sensors and to that end we provide the predictability bounds for the four prominent sensors in the dataset, specifically GSM, WLAN, Bluetooth and acceleration at 15 min. window length. Fig. 3 shows the predictability bounds for the GSM, WLAN, Bluetooth and acceleration/activity sensors. By examining the difference between $\Pi^{\max}$ and Π^{unc} we find that there is considerable gain in knowing the past. From the difference between the Markov model Π^{Markov} and the upper bound, Π^{Max}, we see that knowing more than just the previous state seems to provide considerable benefit.

In the uncorrelated case, H^{unc}, participants show considerable differences in the entropy for all sensors as typically expected in a real population. However, conditioned on the past, the variability is typically lower (except for WLAN) indicating that participants with high entropy have a relative predictable trajectory. This corresponds well with the results observed in [11] for GSM based localization. The Bluetooth data does, as mentioned, contain a considerable fraction of unknown states (not off-states), which will tend to underestimate the true entropy. This means that the bounds are most likely overestimated for Bluetooth.

The GSM, WLAN and Bluetooth sensors are inherently location or proximity oriented sensors, and the estimated distributions all have modes in the high range above 80%, but with noticeable difference in variability. Whereas GSM provides small variability among participants, WLAN seems to provide a much larger difference among participants. This is not surprising since WLAN is considerably

more noisy than GSM and captures a more local and detailed state than GSM. Thus, some participants tend to have a relatively low predictability while others are just as predictable as using GSM. The GSM complexity is on the other hand more similar between mobile phone users. In effect WLAN is most likely the more interesting sensor, but harder to predict, at least using the very detailed representation.

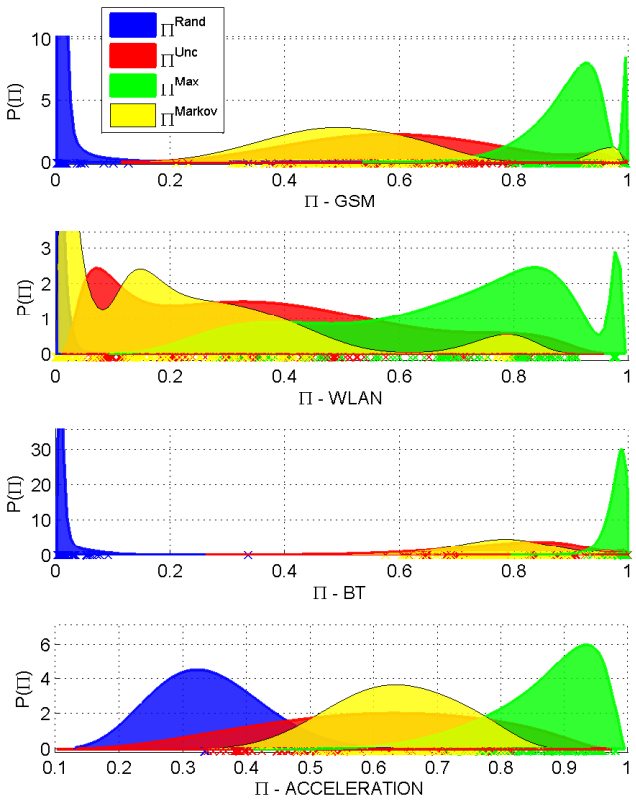


Fig. 3. Detailed representation: Predictability of GSM, WLAN, Bluetooth and Acceleration sensors. Crosses indicate individual participant estimates. The mode at 0.99 and 0.98 in the GSM and WLAN densities are due to participant 3 which is left out in the further analysis for clarity.

4.2 Time Scale

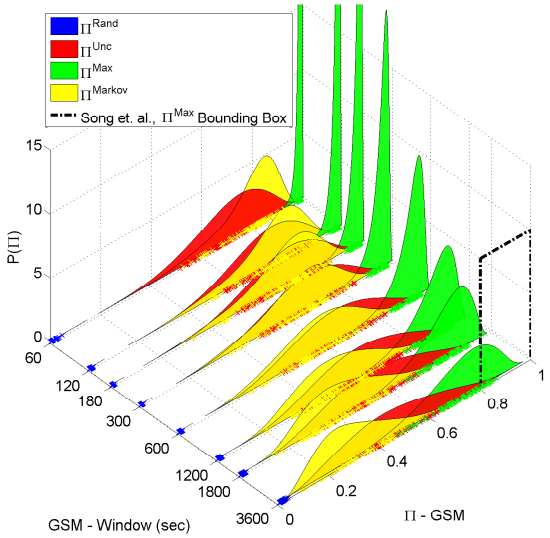
A primary goal in this study is the analysis of the time scales involved in the prediction of location based sensors with the aim to provide support for context-based services and general understanding of human mobility. The focus in this part is thus focused on the GMS and WLAN sensors and the predictability on a wide range of window lengths - and the examination of the optimal scale suggested by the predictive information.

Figure 4 shows how the predictability bounds changes with the window length. Noticeable are the GSM results in Figure 4(a) which are directly comparable with the original results in [11]. The bounding box indicates that the predictability is in the same range, although smaller in our case, possibly due to the more detailed representation utilized here. However, we obtain a similar upper bound at approximately 10 min. scale. This trend towards a high upper bound continues as the scale progresses downwards to 60 seconds.

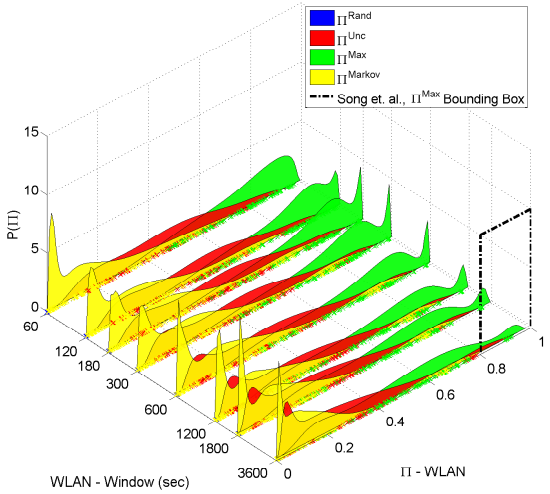
In general there are various fundamental ways why this might happen. First of all, we may simply oversample a constant process leaving the resulting times series highly trivial to predict. The second reason is that the fundamental dependencies are removed when aggregating the cell at long scales and the shorter time scale provides the best representation. A simple way to examine the first options is to look at the predictability suggested by the first order Markov model and determine how far it is from reaching the upper bound. We notice that the despite Π^{Markov} approaching Π^{Max}, there seems to be some non-trivial behavior which is not predicted by the first order model. Not surprisingly this indicates that the first order Markov model is too simple. However, the bound provides a very convenient indication of what a more complex model is able to obtain.

Whereas GSM provides a rather rough indication of mobility and specific location, WLAN cells have quite high location resolution. Examining the time scale for WLAN reveals that the complexity of the problem is very high compared to the GMS case as seen on the pure results obtained by the Markov model. Despite this, we notice that the upper bound is still quite high. This suggests that there is an large unexploited potential in applying a more complex model than for example a first order Markov model. As with the GSM sensor we find that the shorter time scales provides a higher upper bound, and noticeably that the variability among the participants are lower, in effect offering better generalization of the predictability across multiple users.

As we have hinted, the optimal scale time scale for predictability for both GSM and WLAN at small time scales, and to examine the precise scale at which the past offers the most information in predicting the future we consider the predictive information. This is depicted in Figure 5 showing how the predictive information depends on the time scale. We find that the optimal scale is in the 3-4 minute range for both types of sensors. This is our main result and supplements the results in Song et al. [11] who focused on longer time scales (60 minutes). The high predictability at short time scales is of great interest for applications and is "good news" for pro-active services based on predicting human needs and behavior. Furthermore, the fact that the two distinct sensors operating on



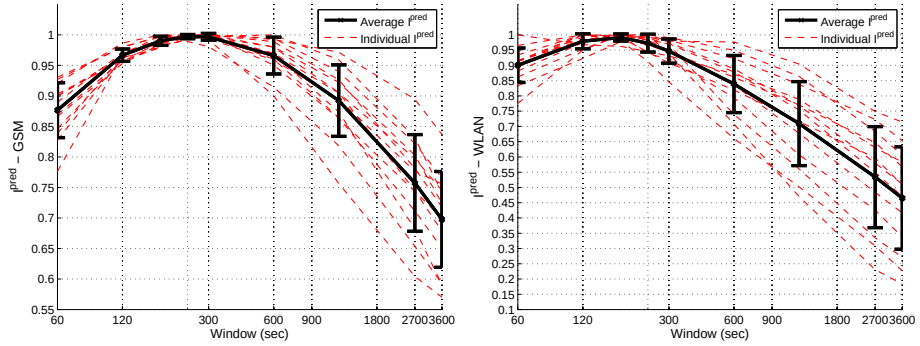
(a) GSM



(b) WLAN

Fig. 4. Predictability vs. window length in sec. (log scale). Notice that participant 3 has been removed from the density estimate due to his/her outlier nature as noticed in Figure 3

different spatial resolution yet still suggest the same optimal temporal scale, indicates that there exists fundamental information at this scale where both the upper bound on GSM and WLAN predictability are quite high. Yet, the exact information available at this scale and implications of this is to be examined in



(a) GSM (b) WLAN

Fig. 5. Predictive Information (normalized) vs. window length (log scale). Participant 3 is left out.

a future analysis, for example using explicit modeling paradigms such as (multi-way) factor analysis and advanced dynamical models.

5 Conclusion

In this paper we described an experimental setup for obtaining so-called *lifelog* data using embedded mobile phones. The resulting dataset offers many possibilities for investigating interesting elements of human behavior.

In the analysis we adopted an implicit modeling approach based on recent information theoretic methods to provide bounds for the prediction one could hope to obtain using explicit modeling. We presented results on the predictability of multiple mobile phone sensors showing that the basic findings in [11] generalizes to more location and proximity based sensors. Specifically, that the gain in knowing the past is significant, which indicates interesting potential for context-aware mobile applications relying on forecasting for example GSM and WLAN associations.

Finally, we showed that the prediction of human mobility generalizes to shorter time scales than the one hour time scale previously studied in [11]. In particular, we showed that the collected GSM and WLAN have the same optimal time scales for prediction, specifically 3-4 minute range. Despite this encouraging result, the exact interpretation and relevance of the patterns at the suggested scale needs further investigation and analysis, for example using explicit modeling.

References

1. MyLifeBits (accessed May 1st 2010). `http://research.microsoft.com/en-us/projects/mylifebits/default.aspx`

2. Bell, G., Gemmell, J.: A digital life. Scientific American 296(3), 5865 (March 2007)
3. Bialek, W., Nemenman, I., Tishby, N.: Predictability, complexity, and learning. Neural Comput. 13(11), 2409–2463 (2001)
4. Eagle, N., (Sandy) Pentland, A.: Reality mining: sensing complex social systems. Personal and Ubiquitous Computing 10(4), 255–268 (2006)
5. Farrahi, K., Gatica-Perez, D.: Discovering human routines from cell phone data with topic models. 2008 12th IEEE International Symposium on Wearable Computers pp. 29–32 (2008)
6. Gao, Y., Kontoyiannis, I., Bienenstock, E.: Estimating the entropy of binary time series: Methodology, some theory and a simulation study. Entropy 10(2), 71–99 (2008)
7. Jensen, B.S., Larsen, J.E., Jensen, K., Larsen, J., Hansen, L.K.: Estimating human predictability from mobile sensor data. In: IEEE International Workshop on Machine Learning for Signal Processing (MLSP 2010) (2010)
8. Kontoyiannis, I., Algoet, P., Suhov, Y.M., Wyner, A.: Nonparametric entropy estimation for stationary process and random fields, with applications to english text. IEEE Transactions on Information Theory 44(3), 1319–1327 (1998)
9. Kwok, R.: Personal technology: Phoning in data. Nature 458(7241), 959–961 (2009)
10. Larsen, J.E., Jensen, K.: Mobile context toolbox - an extensible context framework for s60 mobile phones. In: Proceedings of the 4th European Conference on Smart Sensing and Context (EuroSSC) (2009)
11. Song, C., Qu, Z., Blumm, N., Barabási, A.L.: Limits of predictability in human mobility. Science 327(5968), 1018–1021 (2010)
12. Song, C., Qu, Z., Blumm, N., Barabási, A.L.: Limits of predictability in human mobility - supplementary material (2010)

Discovering Trend-Based Clusters in Spatially Distributed Data Streams

Anna Ciampi, Annalisa Appice, and Donato Malerba

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

Abstract. Many emerging applications are characterized by real-time stream data acquisition through sensors which have geographical locations and/or spatial extents. Streaming prevents from storing all data from the stream and performing multiple scans of the entire data sets as normally done in traditional applications. The drift of data distribution poses additional challenges to the spatio-temporal data mining techniques. We address these challenges for a class of spatio-temporal patterns, called trend-clusters, which combine the semantics of both clusters and trends in spatio-temporal environments. We propose an algorithm to interleave spatial clustering and trend discovery in order to continuously cluster geo-referenced data which vary according to a similar trajectory (trend) in the recent past (window time). An experimental study demonstrates the effectiveness of our algorithm.

1 Introduction

Spatio-temporal data mining has recently gained considerable attention from both the research and practitioner communities. The main reason for this interest is that datasets containing prominent spatial and temporal data elements are growing rapidly due to the daily collection of geo-referenced data through sensor networks.

Despite the significant advances made recently in spatio-temporal data mining, many existing spatio-temporal data analysis approaches take only a static view of the geo-spatial phenomena [7]. These approaches extract a finite set of data points based on user-provided criteria (e.g., the space and time of interest) and address data mining tasks for static spatio-temporal data only. However, a static perspective is inadequate as data often arrive dynamically, continuously and with a drift of the underlying data distribution. On the other hand, a dynamic perspective poses challenges such as avoiding multiple scans of the entire data sets, optimizing memory usage, and mining only the most recent patterns.

The sliding window model [3] is a natural choice to efficiently mine a stream of data where the most recent data observations are considered to be more critical and preferable. This model improves the space/time efficiency of learning by capitalizing on the fact that multiple scans due to pattern evaluation are limited to basic slides.

The sliding-window model was originally defined for a single, non-spatially related data stream source. The main contribution of this work is its extension to the case of a several *spatially distributed* data stream sources. This way, the model can be exploited in a wider range of ubiquitous knowledge applications, which are characterized by both temporal and spatial locality [16].

The scenario we consider is that of a stream composed by sets of observations for a numeric attribute (theme) which are transmitted at consecutive time points by a (variable) number of sources. These stream sources are identified by a progressive number and geo-referenced. By taking into account both the spatial arrangement of the transmitting sources and the temporal arrangement of streamed data, we design a framework to mine a kind of spatio-temporal patterns, called *trend clusters*. These patterns are *spatial clusters* of sources which transmit values whose temporal variation, called *trend polyline*, is similar over the recent window time. We introduce an algorithm, named TRUST (TRend based clUstering algorithm for Spatio-Temporal data stream), for mining these trend-clusters from spatially distributed data streams. The algorithm makes no assumption on the number of spatial clusters as well as on their shape.

The paper is organized as follows. The next section formalizes the problem statement. Section 3 revises related works. Section 4 describes the algorithm and Section 5 reports an experimental study.

2 Problem Statement

In this work, spatio-temporal data are modeled according to the *snapshot* data model [2], where the time domain is linear, absolute and discrete, the spatial domain is field-based and the attribute domains are variable. For every time point of the stream, values collected at this time point form a time-stamped thematic layer. This way, a layer is a collection of geo-referenced values measured for a thematic attribute at the time point. Time-stamped values vary over a continuous space surface which is modeled in the field-based data model [19] as a function: $f : \mathbb{R}^2 \times T \mapsto \textit{Attribute domain}$, where $\mathbb{R}^2$ is an Euclidean representation of the space surface and T is the time line. For the kind of applications considered in this work, it is reasonable to assume that the locations of the transmitting sources are fixed. Therefore, field functions are actually defined over only a finite set of positions, those of active sources. Though finite, field domains can vary over consecutive layers, since the number of transmitting sources can change in time (sources may become inactive and transmit no data for some time interval).

In the stream perspective, thematic layers arrive continuously at consecutive time-points. According to the sliding window schema [9], the stream is broken down into *slides* of p layers arriving in series. A sliding window of size w is composed by w consecutive slides (see Figure 1). Both window and slide sizes are manually defined. In particular, window size depends on the time horizon we are interested in processing. Differently, the slide size should be tuned in order to find the best trade-off between learning time and accuracy. Each time a slide flows in, patterns are generated locally to the slide. Then these local patterns

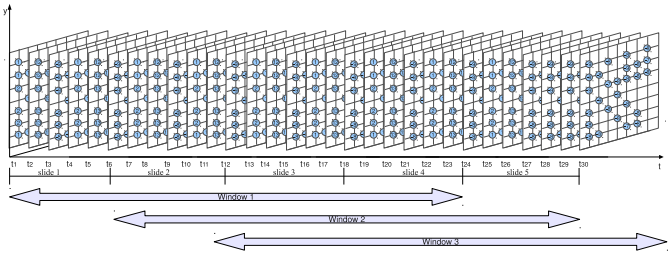


Fig. 1. Sliding windows with $p = 6$ and $w = 4$.

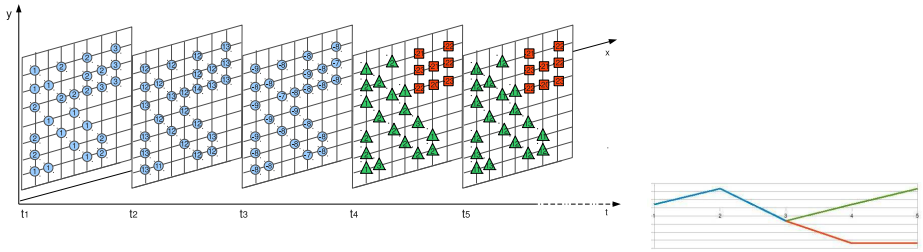


Fig. 2. Trend clusters: the blue cluster groups circle sources which vary according to the blue trajectory over t1, t2 and t3. The red (green) cluster groups squared (triangular) sources which vary according to the red (green) trajectory over t4 and t5. Numbers are the values transmitted by the sources.

are used to approximate the set of patterns in the recent sliding window. This way, multiple scans due to pattern evaluation are limited to slides, thus making the discovery process more efficient both in time and space.

Patterns considered in this work, called *trend clusters*, are the result of *spatial clustering* and *trend discovery* with the sliding window model. Spatial clusters are obtained by processing data in a single time-stamped layer, so that sources in the same cluster model the continuity in space of the measured attribute, while sources in separate clusters model the variation over space [15]. Spatial continuity expresses the similarity between observed values over the spatial organization, which is defined by spatial relations (e.g., distance) between distinct sources. Sources grouped together in spatial clusters of consecutive layers model a spatially local trend for the measured attribute. Trend clusters for an attribute can be represented by colored trajectories over time (see Figure 2).

3 Related Works

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

Data Stream Mining. Guha et al. [10] have presented a constant factor of approximation for the k-Medians algorithm which performs a single pass clustering in data stream. Babcock et al. [4] have extended this algorithm by framing k-Medians algorithm in sliding window model. The limitation of both algorithms is that their results are often spherical clusters. They do not consider that clusters in data streams could be of arbitrary shape and arbitrary number. Aggarwal et al. [1] have proposed a two-phases clustering algorithm, called CluStream, which separates out the clustering process into an online micro-clustering phase and an offine macro-clustering phase. The problem with CluStream is the predefined constant number of micro-clusters. Additionally, since a variant of k-means is adopted to get the final macro-clusters, a "natural" macro-cluster may be split into two parts. To discover arbitrary clusters in data stream and to handle outliers, Cao et al [6] have proposed a density-based clustering algorithm, called DenStream. Finally, unsupervised naive-Bayesian network algorithms which are designed to discover clusters in distributed data streams [17, 5] may be relevant for this work. In fact, a network is here employed to model spatial arrangement of data. Anyway, the kind of pattern we discover, that is, patterns with a polyline trajectory to describe how data vary in time within the cluster, requires a time series based processing of streamed data which is not performed by these naive Bayesian network algorithms.

Spatio-temporal Data Mining. Most of research in spatio-temporal data clustering is focused on the discovery of either trajectory clusters or moving clusters. Trajectory clusters group trajectories of similar shape. Vlachos et al. [20] have firstly defined a Least Common Subsequence distance which is used as a base to apply traditional (partitioning and hierarchical) clustering algorithms to object trajectories. Gaffney and Smyth [8] have proposed a clustering algorithm which models a trajectory as individual sequence of points generated from a regression model. Unsupervised learning is carried out using EM algorithm to cope with the cluster memberships. Nanni and Pedreschi [18] have adapted density-based algorithm to trajectory data by using a distance measure between trajectories. Differently, Lee et al. [13] have proposed a partition-and-group framework for clustering trajectories which partitions a trajectory into a set of segments, and then, groups similar segments into a cluster. Differently, the moving clusters group moving objects in clusters whose identity remains unchanged while cluster location and content may change over time. The key difference 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 object may leave the group or new objects may enter it). A seminal method to discover moving clusters from an history of recorded trajectories has been proposed in Kalnis et al. [12]. Spatial clusters are discovered at each snapshot by resorting to a static density-based clustering algorithm and results are then combined into a set of moving clusters. Li et al. [14] have proposed to exploit the micro-clustering originally proposed by Zhang et al. in [21] and extend it to moving micro-clustering. In this work, a moving micro-cluster denotes a group of objects that are not only close to each other at current time, but also likely to move together for a while.

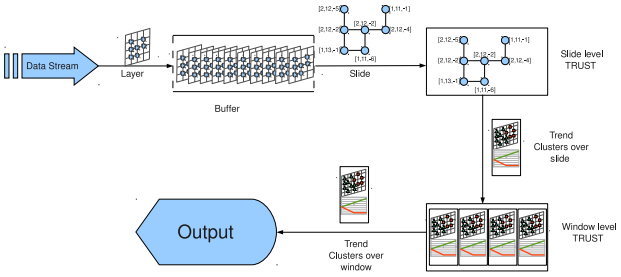


Fig. 3. The framework of TRUST.

4 The Algorithm

The framework of TRUST is reported in Figure 3. A buffer continuously consumes thematic layers and pours them slide-by-slide into the TRUST system. TRUST operations consist in: (1) buffering slide layers in a graph-based synopsis data structure, (2) discovering trend-clusters over the slide time (slide-level clustering), and then (3) approximating trend clusters by combining the slide-level trend cluster sets (window-level clustering). After a slide goes through TRUST, slide layers are discarded, while the set of trend cluster discovered over the slide is maintained in main memory for the window time. For each slide, the number of discovered trend clusters ranges between 1 (i.e., sources are all grouped in a single cluster) and the number of sources (one cluster for each source). Due to the sliding window model, once the set of trend clusters is locally discovered over the last income slide, the oldest set is discarded. The number p of layers in a slide, the number w of the slides composing the sliding window, the value-similarity threshold δ, the slide-level trend continuity threshold θ and the window-level trend continuity threshold ϵ are given before TRUST starts.

4.1 Buffer Synopsis Structure

Since TRUST is in charge of discovering trend clusters over the sliding windows of a data stream, it needs to maintain on-line a representation of the spatial organization which arises in the layers of the current sliding window.

As reported in [15], the spatial organization of a layer can be modeled by means of a graph $G(N, E)$. In this work, N is the set of nodes (sources) which transmit signals over at least one slide of the sliding window. E is a binary (spatial) relation between nodes, $E \subseteq \{\langle u, v\rangle | u, v \in N\}$ (e.g., an edge $\langle u, v\rangle \in E$ connects the nodes u and v iff the Euclidean distance between the corresponding sources is less than a threshold). Each node in N is assigned with w binary labels which describe the active/inactive state of the node at each slide of the sliding window. A node is active at a slide if it is active in at least one layer in the

slide, inactive otherwise. The nodes of G which are active at the current slide defines the structure of the data synopsis where the layers are buffered for the time of the slide-level clustering phase. Similarly, the edges of E which connect nodes which are active on the slide identify the spatially close sources which are the only candidates to be grouped in the same cluster during the slide-level clustering phase. The entire edge set E is used during the window-level clustering phase to approximate the set of trend clusters for the entire window.

The graph G is constructed from scratch with the first slide which flows in the stream. After that, the structure of G is updated each time a new slide flows in the stream. The update operations consist in: (1) adding a new node (and the edges) to N (and E) in order to map a transmitting source which is firstly monitored in the sliding window, the state of this node is active over the current slide, inactive over the past $w-1$ slides of the sliding window; (2) updating an existing node f N by discarding the oldest state label and assigning an active label if the source transmits signals in the current slide (inactive otherwise); and (3) removing a node (and the edges) from N (and E) if the node was labeled as inactive over each slide of the sliding window.

Let us denote: N_A as the set of nodes which belong to N and are active in the current slide, and E_A as the set of edges which belong to E and connect active nodes in the current slide. $G_A(N_A, E_A)$ represents the graph structure of the synopsis where the lastly income slide layers are temporally stored for the slide-level clustering phase. For each active node $u \in N_A$, a p-sized bucket is stored in the synopsis data structure. This bucket is denoted as B_u and it stores the p values streamed by the corresponding source in the slide time (see Figure 4). This way, the slot $B_u[l]$ is the value buffered in u by the l^{th} layer which arrives in the slide. It is noteworthy that there are two cases in which missing values may be stored in a bucket. In the former case, a source, which has already been active in the past, does not transmit the signal for one or more layers in the current slide (e.g., the sensor is gone down or the transmission is not in time), the missing value is replaced by the lastly observed value for the source. In the latter case, a source transmits the first signal at a layer of the current slide, missing values which precede the first transmission are replaced with this firstly transmitted value.

4.2 Slide-level trend cluster Discovery

Before presenting how the slide-level trend cluster discovery is performed, we introduce some preliminary definitions.

Definition 1 (δ-close measure ψ_δ). *Let X be a continuous theme with domain $[\alpha, \beta]$ and δ be a real value in $[0,1]$. The δ-close measure is a function $\psi_\delta \colon X \times X \mapsto \{0,1\}$ such that* $\psi_\delta(x_1, x_2) = \begin{cases} 1 & \frac{\|x_1 - x_2\|_1}{\beta - \alpha} \leq \delta \\ 0 & otherwise \end{cases}$.

Based on Definition 1, we define the E_δ^θ close relation between the edged nodes of G.

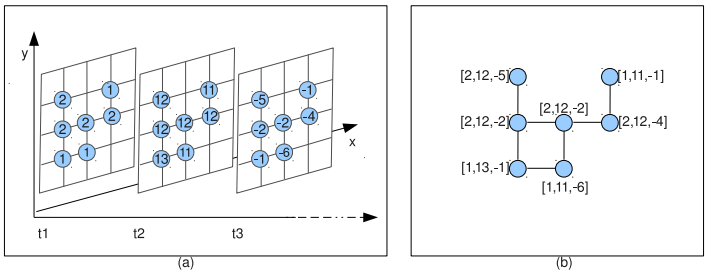


Fig. 4. The graph synopsis data structure (b) where layers (a) are buffered with $p = 3$.

Definition 2 (E_δ^θ close relation). *Let δ be a real value in $[0,1]$ and θ be a real value in $[0,1]$; E_δ^θ is the binary relation between the edged nodes of G_A ($E_\delta^\theta \subseteq E_A$) which is defined as $E_\delta^\theta = \{\langle u,v\rangle \in E | \sum_{i=1}^{p} \psi_\delta(B_u[i], B_v[i]) \geq \theta \times p\}$, where $B_u[i]$ ($B_v[i]$) is the i^{th} slot value in the bucket B_u (B_v).*

We use the E_δ^θ close relation to define the E_δ^θ-connectivity in G.

Definition 3 (E_δ^θ-connected). *A node u is E_δ^θ-connected to a node v ($u, v \in N_A$), with respect to E_δ^θ, iff: (i) $\langle u,v\rangle \in E_\delta^\theta$,* (direct connectivity) *(ii) or $\exists w \in N_A$ such that $\langle u,w\rangle \in E_\delta^\theta$ and w is E_δ^θ-connected from v* (undirect connectivity).

Finally, we define the function of *δ-homogeneity* over G_A.

Definition 4 (δ-homogeneity). *Let δ be a real value in $[0,1]$. The function δ-homogeneity $: 2^N \mapsto \{true, false\}$ is defined as:*

$$\delta - homogeneity(N_i) = \begin{cases} true & \forall u,v \in N_i,\ \frac{\|\eta(B_u) - \eta(B_v)\|_1}{\beta - \alpha} \leq \delta \\ false & otherwise \end{cases}.$$

In Definition 4, the function $\eta\colon 2^X \mapsto X$ returns the middle point (median) over a sequence of values in X. The choice of the median is motivated by the fact that it is robust, while the mean would be influenced by outliers. Finally, a trend cluster is defined as follows.

Definition 5 (trend cluster). *A trend cluster is the pair $C \Leftrightarrow P$ such that:*

1. *$C \subseteq N$ satisfies the following properties: (a) $\forall u,v \in C$: u E_δ^θ-connected to v, and (b) C is a δ-homogeneous node set , i.e., δ-homogeneity(C)=true;*
2. *$P = \langle \eta(C|_1), \eta(C|_2) \ldots, \eta(C|_p)\rangle$ where $C|_l$ is the set $\{B_u[l] | u \in C\}$ and $\eta(\cdot)$ is the median function.*

Note that, in Definition 5, the E_δ^θ-connectivity guarantees the (spatial) continuity of polylines which are associated to the nodes grouped in a cluster at

Algorithm 1 Slide-level discovery: $G_A(N_A, E_A) \mapsto \Gamma$

– *Main routine*

1: $\Gamma = \oslash$
2: **for all** $u \in N_A$ **do**
3: **if** u is $UNCLASSIFIED$ **then**
4: $C \leftarrow \{u\}$
5: $C \leftarrow expandCluster(C, G_A, u)$
6: $P \leftarrow \langle \eta(C,1), \eta(C,2), \ldots, \eta(C,p) \rangle$
7: $\Gamma \leftarrow \Gamma \cup \{C \Leftrightarrow P\}$
8: **end if**
9: **end for**

– *expandCluster* (C, G_A, u)

1: $N_\theta^\delta(u) \leftarrow E_\delta^\theta neighborhood(u, G_A)$
2: **if** δ-homogeneity$(C \cup N_\delta^\theta(u))$ **then**
3: $C \leftarrow C \cup N_\delta^\theta(u)$
4: **for all** $n \in N_\delta^\theta(u)$ **do**
5: $C \leftarrow expandCluster(C, G_A, n)$
6: **end for**
7: **else**
8: **for all** $n \in N_\delta^\theta(u)$ **do**
9: **if** δ-homogeneity$(C \cup \{n\})$ **then**
10: $C \leftarrow C \cup \{n\}$
11: $C \leftarrow expandCluster(C, G_A, n)$
12: **end if**
13: **end for**
14: **end if**

each layer. The condition on $\delta-$homogeneity is added to guarantee that the entire cluster evolves with a *single* close polyline that is reasonably approximated with P. Closeness depends on δ and θ. Since a cluster is the node set of a sub-graph extracted from G by satisfying the conditions of E_δ^θ-connectivity and δ-homogeneity, TRUST resorts to a neighborhood-based graph partitioning algorithm-like in order to discover the trend clusters over slide.

The top-level description of the algorithm is in Algorithm 1. The key idea is to exploit the construction of node neighborhood based on the E_δ^θ close relation (namely E_δ^θ-neighborhood) and to grow clusters by merging partially overlapping E_δ^θ neighborhoods only if the resulting cluster is a δ-homogeneous node set. The definition of the E_δ^θ-neighborhood of a node u in G is reported below.

Definition 6 (E_δ^θ-neighborhood). *Let u be a node, the neighborhood $N_\delta^\theta(u)$ of u is defined as $N_\delta^\theta(u) = \{v | \langle u, v \rangle \in E_\delta^\theta \quad \wedge UNCLASSIFIED(v)\}$, where $UNCLUSSIFIED(v)$ means that v is not assigned to any cluster.*

In the *main routine* of Algorithm 1, a novel empty cluster C is created for a node $u \in N$ which is currently $UNCLASSIFIED$. C is firstly grown with u, then $expandCluster(C, u)$ grows C by using u as seed. In particular, $expandCluster(C, u)$ evaluates the property of $\delta - homogeneity$ for the node

set $C \cup N_\delta^\theta(u)$ (see the call of $\delta - homogeneity(\cdot)$ function in Algorithm 1). If $C \cup N_\delta^\theta(u)$ is a δ-homogeneous node set, C is grown with $N_\delta^\theta(u)$. Otherwise, the addition of each neighbor $n \in N_\delta^\theta(u)$ is evaluated node-by-node. A neighbor is individually added to C only if the output cluster remains a $\delta-$ homogeneous node set. Each time C changes (by either adding an entire neighborhood or only few neighbors), $expandCluster(\cdot,\cdot)$ is recursively called to further grow C by considering the new clustered nodes as candidate seed of the expansion (see the recursive call of $expandCluster(\cdot,\cdot)$ in Algorithm 1).

When a cluster C is completely constructed (no further node can be added to), the polyline P, which describes how the clustered nodes evolve over the slide time, is built. P is the sequence of p points, that is defined as $P = \langle 1, \eta(C,1)\rangle, \langle 2, \eta(C,2)\rangle, \ldots, \langle p, \eta(C,p)\rangle$, where each point $\langle l, \eta(C,l)\rangle$ is the representative of the behavior of the cluster C over the l^{th} layer in the buffered slide. In Algorithm 1, the function $\eta(C,l)$ (that is called with argument $l = 1,\ldots,p$) returns the median of the values stored in the l^{th} slots for the nodes grouped in C.

Finally, each trend cluster $C \Leftrightarrow P$ is added to the cluster set Γ. Once all the nodes have been classified in a cluster, values in the buckets are discarded and the algorithm stops by returning Γ as output of the slide-level discovery process. Note that Γ can be intended as a reasonable summarization of the slide layers.

4.3 Window-level trend cluster Discovery

The discovery of trend clusters over a window capitalizes on the trend cluster sets, denoted as $\Gamma_1,\ldots,\Gamma_w$, which have been locally discovered for the slides of W. Indeed, while past slide layers are discarded, the discovered trend cluster sets are maintained for the window time. This way, the window-based discovery analyzes a reduced set of data. The window level clustering bases on the consideration that nodes, which are repeatedly grouped together in a cluster for each slide of the window, map sources which are close in space and which evolve with a close polyline over the entire window time. Based upon this consideration, the window-level component of TRUST determines the trend clusters over the window by clustering the spatially close sources which are continuously clustered together over the slides of the window. Clustering is performed based on the binary cluster relation denoted as $E_\epsilon^{\Gamma_1,\ldots,\Gamma_w}$ which is defined below.

Definition 7 ($E_\epsilon^{\Gamma_1,\ldots,\Gamma_w}$ window cluster relation). *Let ϵ be a real value in $[0,1]$, $E_\epsilon^{\Gamma_1,\ldots,\Gamma_w}$ is defined over E as $E_\epsilon^{\Gamma_1,\ldots,\Gamma_w} = \{\langle u,v\rangle \in E | \sum_{i=1}^{w} clustered(u,v,\Gamma_i) \geq \epsilon \times w\}$ with $clustered(u,v,\Gamma_i) = \begin{cases} 1 & \exists (C \Leftrightarrow P) \in \Gamma_i : u,v \in C \\ 0 & otherwise \end{cases}$.*

The binary window cluster relation $E_\epsilon^{\Gamma_1,\ldots,\Gamma_w}$ identifies each pair of edged nodes of G which are clustered together over at least $\epsilon \times w$ slides of W. Hence, clusters over window are obtained by partitioning the graph $G(N,E)$ in completely connected sub-graphs according to the binary window cluster relation

Algorithm 2 Window-level discovery: $\Gamma_1, \Gamma_2, \ldots, \Gamma_w, G(E,N) \mapsto \Gamma$

– *Main routine*

1: $E_\epsilon^{\Gamma_1,\ldots,\Gamma_w} \leftarrow determineWindowClusterRelation(\Gamma_1, \Gamma_2, \ldots, \Gamma_w, E)$
2: $\Gamma \leftarrow \oslash$
3: **for all** $u \in N$ **do**
4: **if** u is $UNCLASSIFIED$ **then**
5: $C \leftarrow \{u\}$
6: $C \leftarrow expandCluster(C, E_\epsilon^{\Gamma_1,\ldots,\Gamma_w}, u)$
7: $P \leftarrow polyline(C, \{\Gamma_1, \ldots, \Gamma_w\})$
8: $\Gamma \leftarrow \Gamma \cup \{C \Leftrightarrow P\}$
9: **end if**
10: **end for**

– *expandCluster*$(C, E_\epsilon^{\Gamma_1,\ldots,\Gamma_w}, u)$

1: **for all** $\langle u, v\rangle \in E_\epsilon^{\Gamma_1,\ldots,\Gamma_w}$ and v is $UNCLUSSIFIED$ **do**
2: $C \leftarrow C \cup \{v\}$
3: $C \leftarrow expandCluster(C, E_\epsilon^{\Gamma_1,\ldots,\Gamma_w}, v)$
4: **end for**

$E_\epsilon^{\Gamma_1,\ldots,\Gamma_w}$ (see Algorithm 2). Let C be a cluster over window W, C is assigned with the trend polyline P and $C \Leftrightarrow P$ is output as a trend cluster over W. P is built by sequencing the points of the polylines which are associated, slide by slide, to the clusters which include C (see the call of function $polyline(\cdot,\cdot)$ in Algorithm 2). Formally, $P = P^1, P^2, \ldots, P^w$ where:

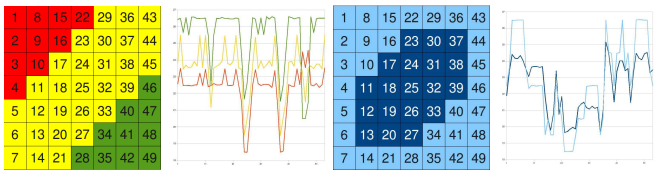
$$P^j = \begin{cases} P_i^j & \exists (C_i^j \Leftrightarrow P_i^j) \in \Gamma_j : C \subseteq C_i^j \\ \text{null} & \text{otherwise} \end{cases}. \tag{1}$$

5 Experiments

We evaluate TRUST with a synthetic data stream and two real data streams. Experiments evaluate accuracy and number of discovered patterns and the efficiency of learning. As a measure of the accuracy of the trend cluster set Γ, we use the average absolute percentage error ($MAPE$), a statistics that is widely used in time series [11]. MAPE computes the absolute error that is performed when the polylines of Γ are used to fit data windowed in W. Formally,

$$MAPE(\Gamma, W) = \sum_{u \in N} mape(\Gamma, u|_W)/|N|,$$

where N denotes the source set over which the stream is transmitted over the window W and $u|_W$ denotes the series of $p \times w$ values which are streamed from the source $u \in N$ over the window W. $mape(\Gamma, u|_W)$ is the mean absolute percentage error which measures how the polyline P of the trend cluster $C \Leftrightarrow P$ fits real values streamed in the $u|_W$ (with $u \in C$). It expresses accuracy as a percentage, and is defined by $mape(\Gamma, u|_W) = \frac{1}{p \times w} \sum_{i=1}^{p \times w} \left\| \frac{(u[i|_W]) - P[i]}{(u[i|_W])} \right\|_1$ with $(C \Leftrightarrow P) \in \Gamma$ and $u \in C$.



(a) (b)

Fig. 5. The trend cluster configurations according to the synthetic data are generated.

Synthetic data stream. As a synthetic data stream we have considered the stream transmitted by 49 sources which are distributed over a squared 7×7 grid. A source is close to the neighbor sources which are located in the grid cells around the source (see Figure 5). Let X be the thematic attribute of the stream, X ranges in $[18, 27]$. The stream is obtained by sequencing 54 layers of measurements of X which are generated according to the trend based cluster configuration reported in Figure 5.a. and 54 layers which are generated according to the trend based cluster configuration reported in Figure 5.b. Each source measures values which evolve according to the polyline representing the trend of the corresponding cluster. For each layer, for each source the polyline value is noised with an error generated according to the Normal distribution N(0,1). Experiments are run with $p = 9$, $w = 6$, $\delta = 10\%[27 - 18] = 0.9$, $\theta = 0.8$ and $\epsilon = 1$. θ is set to a value different from 1 in order to take into account variation in data due to the noise. This way TRUST processes twelve slides and discovers trend clusters over seven consecutive sliding windows. The cluster configurations discovered for each sliding window are reported in Figure 6. As expected, the clustering results show that TRUST is able to exactly detect the cluster configuration that is defined in (a) over the first window (slides 1-6) and the cluster configuration that is defined in (b) over the last window (slides 6-12) of the processed stream. The results of clustering over intermediate sliding windows correctly reveal the concept drift from the configuration (a) toward the configuration (b). Indeed, to adapt the discovered cluster configuration to the effective evolution trend of the windowed data, TRUST moves the sources 5, 6, 7, 14, 21, 29, 36, 43, 44 and 45 in two new spatial clusters. This way, TRUST still identifies groups of spatially contiguous sources whose trend is approximately the same over the window under consideration. It is noteworthy that 5, 6, 7, 14 and 21 are grouped in one spatial cluster, while 29, 36, 43, 44 and 45 are grouped in a separate cluster over the intermediated sliding windows (slides 2-8, slides 3-9, slides 4-10, slides 5-11) due to the spatial discontinuity of sources, while the plot of the trend associated to these spatial clusters show two polylines which are overlapping. In Table 1, the mean absolute percentage error ($MAPE$) of trend cluster sets discovered over the sliding windows of data stream is reported. $MAPE$ provides an estimate of the accuracy of cluster polylines in fitting windowed data. The observed value of $MAPE$ is always low (it is less than 0.03 for each window), thus confirming that the trend discovery is accurate.

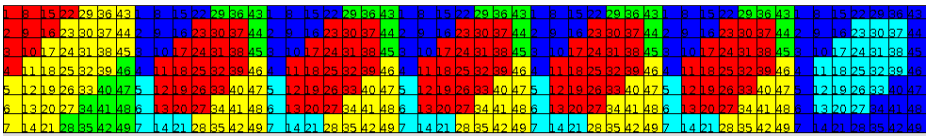


Fig. 6. The clusters discovered by TRUST over each sliding window of the synthetic data stream ($p = 9$, $w = 6$, $\delta = 10\%(27 - 18)$, $\theta = 0.8$ and $\epsilon = 1$).

Table 1. *MAPE* of trend cluster sets over sliding windows of processed data stream.

error	**Sliding windows**						
	[1-6]	[2-7]	[3-8]	[4-9]	[5-10]	[6-11]	[7-12]
MAPE	0.01885	0.01921	0.01975	0.02083	0.02229	0.02267	0.02290

Intel Berkley Lab data stream. Intel Lab data stream[1] contains real information collected from 54 sensors deployed in the Intel Berkeley Research lab between February 28th and April 5th. A sensor is considered to be close to each other sensor into the six meters range. The sensors which we consider in this experiment have collected timestamped temperature values once every 31 seconds (epoch). There are several epochs where no temperature value is transmitted by one or more sensors (missing values). We have processed the entire stream that is 2.3 about million readings (about 65000 layers) collected from these sensors in 73541 millisecs by setting $p = 20$ and $w = 5$, $\theta = 1$ and $\epsilon = 1$. Due to the space limitation, we describe only the trend cluster configurations which have been discovered by processing a subset of 5000 layers. Experiments are run with $\delta = 10\%[\beta - \alpha]$. Due to the presence of several outlier values, we use a box plot to derive a reasonable representation of the data range $[\alpha, \beta]$. The groups of streamed values collected over an initial calibration time are depicted through their five-number summaries (box plot): the smallest observation (sample minimum), lower quartile (Q_1), median (Q_2), upper quartile (Q_3), and largest observation (sample maximum). This way we derive $\alpha = Q_1 - 1.5 * (Q_3 - Q_1) = 10.405$ and $\beta = Q_3 + 1.5 * (Q_3 - Q_1) = 37.085$. Hence, $\delta = 2.668$. To evaluate the effectiveness of sliding window model in the stream environment, we compare each trend cluster set obtained with $p = 20$ and $w = 5$ with the cluster results obtained by learning the full widowed data at once ($p = 100$ and $w = 1$). Elapsed time of the learning phase with sliding windows is reported in Figure 7.a. The comparison between $p = 20$ $w = 5$ and $p = 100$ $w = 1$ is plotted in Figure 7.b. We consider only the windows which cover the same portion of data stream (e.g., slides 1-5, 6-10, and so on). Since $\theta = 1$ and $\epsilon = 1$ the same trend clusters are discovered in both configurations, while the elapsed time is different due to the use of sliding window model. We can observe that TRUST capitalizes on the slide-based discovery to analyze a reduced set of data (slide), thus decreasing the time of learning in the online discovery process without affecting number and trend of discovered clusters. This is an empirical confirmation of effectiveness

[1] http://db.csail.mit.edu/ labdata/labdata.html

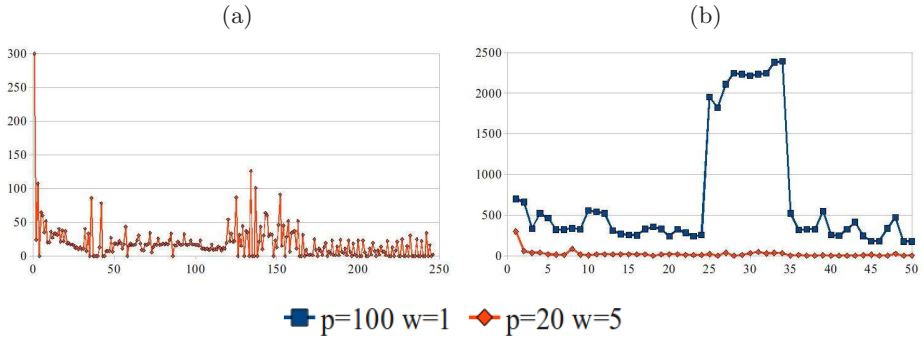


Fig. 7. Elapsed time (ms) with (a) $p = 20$ $w = 5$ over the sliding windows, (b) $p = 20$ $w = 5$ vs. $p = 100$ $w = 1$ over windows which cover same data portion. $\theta = 1$ $\epsilon = 1$.

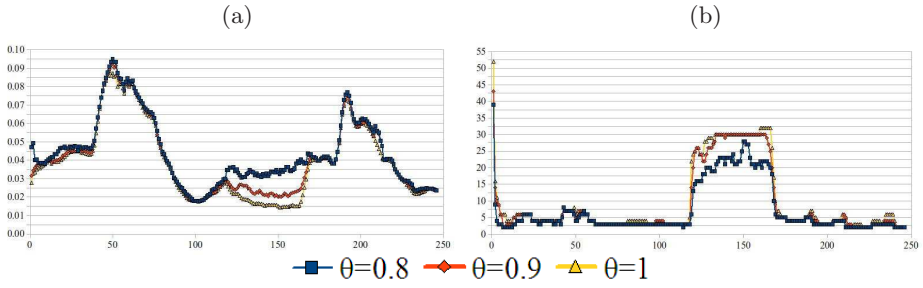


Fig. 8. (a) MAPE and (b) number of clusters with $p = 20$ and $w = 5$.

of sliding window model to improve efficiency of mining trend-based patterns in a stream. Further experiments have been performed to evaluate dependence of TRUST from θ. The number of clusters as well as the MAPE for consecutive sliding-windows are plotted in Figure 8.a-b by varying θ between 0.8, 0.9 and 1. The number of clusters shows that TRUST is able to group sources over windows thus summarizing windowed data by means of the polylines which are associated to the clusters (the number of discovered clusters is generally less than 10 with a pick to 32 in the central plotted windows). As expected, we observe that by relaxing the threshold of the trend continuity over slide (θ), the number of clusters decreases and, consequently, also the accuracy of fitting decreases. The fitting capability is always good enough with an error that is at worst 0.1. The issue is to find the trade-off between summarization degree (number of clusters) and fitting accuracy (MAPE). In this work, we do not consider $\epsilon < 1$ since we intend to detect window-time continuous trend clusters.

South American Climate data stream. South American climate data stream[2] contains monthly-mean air temperature values recorded between 1960 and 1990 over a 0.5 degree by 0.5 degree of latitude/longitude grid of South America, where

[2] http://climate.geog.udel.edu/∼climate/html_pages/archive.html

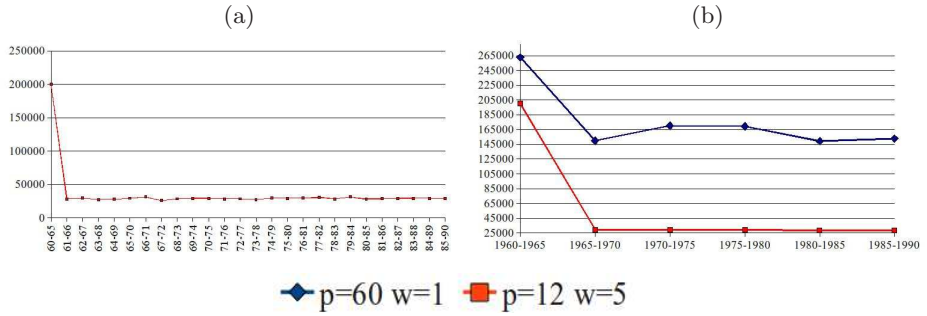


Fig. 9. Elapsed time (ms) with (a) $p = 12$ $w = 5$ over the sliding windows, (b) $p = 12$ $w = 5$ vs. $p = 60$ $w = 1$ over windows which cover same data portion. $\theta = 1$ $\epsilon = 1$.

the grid nodes are centered on 0.25 degree for a total of 6447 node sources. A source is close to the neighbor sources which are located in the grid cells around the source. Experiments are run with $\delta = 10\%[\beta - \alpha]$. We use the box plot to group streamed temperature values and to determine $\alpha = 6.925$, $\beta = 37.125$ and, hence, $\delta = 3.02$.

The elapsed time of the learning process is reported in Figure 9 for two parameter settings: a) $p = 12$ (one year) and $w = 5$; b) $p = 60$ and $w = 1$. Windows cover the same portion of data stream. Also in this stream, we observe that TRUST capitalizes on the slide-level discovery to improve efficiency of learning. Although the number of sources is significantly high, learning is efficiently enough to run on-line. Indeed, after the first window, where five slides have to be processed before being able to output clusters over the window, the elapsed time to learn each single slide and derive the clusters over the window is less than 30 secs.

We also evaluate the effectiveness of trend clusters discovered by varying θ between 0.8, 0.9 and 1. The number of clusters as well as the MAPE values for consecutive sliding-windows are plotted in Figure 10.a-b. We observe that the number of clusters is greatly lower than 6447 (at worst 4 clusters over each window), thus TRUST is effective in summarizing data. Additionally, MAPE value is low (less than 0.06 independently on θ), this confirms that also in this stream the summarization by means of the discovered trend clusters is accurate in fitting the real windowed data. A final consideration concerns θ, by relaxing which, the number of clusters decreases only over few windows and in these cases also the accuracy of fitting slightly decreases.

Some conclusions can be drawn from this empirical study. TRUST is effective in discovering accurate trend-based clusters as we proved with the analysis of MAPE. TRUST is scalable as we proved with the analysis of data streams generated by both a low number of sources (synthetic data and Berkley Lab data) and an high number of sources (South American Climate data). Additionally, TRUST is able to operate in real world situations and cope with possible variation in the sources configurations over consecutive layers (Berkley Lab data).

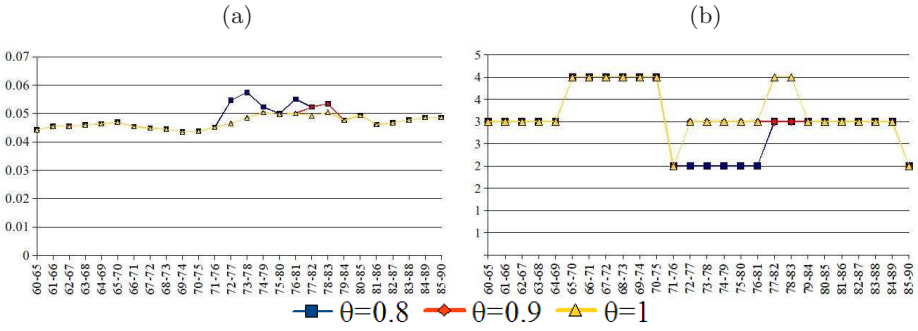


Fig. 10. (a) MAPE and (b) number of clusters ($p = 12$ $w = 5$).

Final considerations concern the utility of patterns that TRUST discovers. Our basic consideration is that trends provide useful information in spatially distributed data stream, where knowing how spatially clustered values vary in time can help drawing useful conclusions. For instance, in weather monitoring, it is essential to know how temperatures evolve (increasing or decreasing) over regions of the Earth and how the shape of these regions changes in time. Additionally, trend-based representation of spatially distributed data stream is considered close to the human intuition. Finally, trend clusters can be used for summarizing spatio-temporal data and subsequent storing in data warehouses.

6 Conclusions

The paper presents TRUST an algorithm to retrieve groups of spatially continuous geo-referenced data which vary according to a similar trend polyline in the recent window past. As future work we plan to investigate how the algorithm depends on the order of analyzing the geo-referenced streamed values and we intend to study criteria to suggest the best order of evaluation.

Acknowledgments

This work is supported by the Strategic Project PS121: "Telecommunication Facilities and Wireless Sensor Networks in Emergency Management".

References

1. C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *VLDB 2003*, pages 81–92, 2003.
2. C. Armenakis. Estimation and organization of spatio-temporal data. In *GIS*, 1992.
3. B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *PODS 2002*, pages 1–16. ACM, 2002.

4. B. Babcock, M. Datar, R. Motwani, and L. O'Callaghan. Maintaining variance and k-medians over data stream windows. In *PODS 2003*, pages 234–243. ACM, 2003.
5. K. Bhaduri, K. D. K. Sivakumar, H. Kargupta, R. Wolff, and R. Chen. Algorithms for distributed data mining. In *Data Streams: Models and Algorithms (book chapter)*, volume 31, pages 309–334. Springer-Verlag, 2007.
6. F. Cao, M. Ester, W. Qian, and A. Zhou. Density-based clustering over an evolving data stream with noise. In J. Ghosh, D. Lambert, D. B. Skillicorn, and J. Srivastava, editors, *SIAM SDM 2006*, 2006.
7. W. Chang, D. Zeng, and H. Chen. A stack-based prospective spatio-temporal data analysis approach. *Decis. Support Syst.*, 45(4):697–713, 2008.
8. S. Gaffney and P. Smyth. Trajectory clustering with mixtures of regression models. In *KDD 1999*, pages 63–72. ACM, 1999.
9. V. Ganti, J. Gehrke, and R. Ramakrishnan. Mining data streams under block evolution. *SIGKDD Explorations*, 3(2):1–10, 2002.
10. S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. In *FOCS*, pages 359–366, 2000.
11. R. Hyndman and A. B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688, 2006.
12. P. Kalnis, N. Mamoulis, and S. Bakiras. On discovering moving clusters in spatio-temporal data. In C. Bauzer Medeiros et al., editor, *SSTD 2005*, volume 3633 of *Lecture Notes in Computer Science*, pages 364–381. Springer-Verlag, 2005.
13. J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory clustering: a partition-and-group framework. In *SIGMOD 2007*, pages 593–604. ACM, 2007.
14. Y. Li, J. Han, and J. Yang. Clustering moving objects. In *KDD 2004*, pages 617–622, New York, NY, USA, 2004. ACM.
15. D. Malerba, A. Appice, A. Varlaro, and A. Lanza. Spatial clustering of structured objects. In S. Kramer and B. Pfahringer, editors, *ILP 2005*, volume 3625 of *Lecture Notes in Computer Science*, pages 227–245. Springer-Verlag, 2005.
16. M. May, B. Berendt, A. Cornujols, J. Gama, F. Giannotti, A. Hotho, D. Malerba, E. Menesalvas, K. Morik, R. Pedersen, L. Saitta, Y. Saygin, A. Schuster, and K. Vanhoof. Research challenges in ubiquitous knowledge discovery. In *Next Generation of Data Mining*. Chapman and Hall/CRC, 1 edition, 2008.
17. R. Munro and S. Chawla. An integrated approach to mining data streams. In *Technical Report, University of Sydney. School of Information Technologies*, 2004.
18. M. Nanni and D. Pedreschi. Time-focused clustering of trajectories of moving objects. *J. Intell. Inf. Syst.*, 27(3):267–289, 2006.
19. S. Shekhar and S. Chawla. *Spatial databases: A tour*. Prentice Hall, 2003.
20. M. Vlachos, D. Gunopoulos, and G. Kollios. Discovering similar multidimensional trajectories. In *ICDE 2002*, page 673. IEEE Computer Society, 2002.
21. T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. *SIGMOD Rec.*, 25(2):103–114, 1996.