Stephan Sigg

Development of a novel context prediction algorithm and analysis of context prediction schemes

kassel university press

Die vorliegende Arbeit wurde vom Fachbereich Elektrotechnik/Informatik der Universität Kassel als Dissertation zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.) angenommen.

Erster Gutachter: Univ.-Prof. Dr.-Ing. Klaus David
Zweiter Gutachter: Univ.-Prof. Dr. Alois Ferscha

Tag der mündlichen Prüfung 25. Februar 2008

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar

Zugl.: Kassel, Univ., Diss. 2008
ISBN 978-3-89958-392-2
URN: urn:nbn:de:0002-3929

www.upress.uni-kassel.de

Druck und Verarbeitung: Unidruckerei der Universität Kassel
Printed in Germany

Abstract

Context-awareness is an area in which we strive to improve the environment-application interaction by the consideration of environmental stimuli and with focus on the application's reaction. The aim is to extend the knowledge of applications on the requirements of a given situation beyond explicit inputs. This approach is motivated by the general observation that a given environment greatly impacts the intention with which an application is executed. Consider, for example, the ringing profile of a mobile phone. When a user is in a meeting, she will prefer a silent mode while in between two meetings she might be willing to be notified of incoming calls by a ring tone. The preference of the ringing profile of the user is situation dependent. In context-awareness, the environmental situation is recorded with the help of sensors. One of the vital parts is then to extract valuable information from this knowledge. This extracted information helps to fine-tune the reaction to the user needs in an ever more advanced manner.

Context-awareness is a broad field that can be further partitioned into sub-disciplines that themselves constitute interesting research topics. A context-aware application might be aware of its present, past or future context. Most work is carried out taking present or past context into consideration. A research branch that is by far less intensively studied is the derivation and utilisation of future context. The latter research topic is commonly referred to as context prediction.

In this thesis we investigate issues related to the inference of future context and to the utilisation of past and present contexts in this setting. We discuss the context prediction task and identify chances and challenges that derive from it. This discussion leads to a definition of the context prediction task.

The context prediction task might be completed by various context prediction schemes. Dependent on the amount of pre-processing or context abstraction applied to context data, we distinguish between various context prediction schemes. We study context prediction schemes for their prediction accuracy and provide guidelines to application designers as to

which prediction scheme to utilise in which situation. Analytical results obtained in these studies are confirmed in simulations in several application domains.

For context prediction to be applied widely in various application domains, standardised architectures might foster this development. However, only a few approaches to context prediction architectures have been considered yet. These proposals are static in regard to the context abstraction level. We develop a novel architecture for context prediction approaches that is applicable to arbitrary context abstraction levels. This architecture also allows distribution of components to various interconnected devices in a mobile ubiquitous computing environment.

Another aspect of context prediction is provided by algorithms for context prediction. For context prediction algorithms, we review the state of the art and discuss strengths and weaknesses of these algorithms. On the basis of requirements identified for algorithms in context prediction scenarios, we are able identify those approaches that are best suited for context prediction tasks. A novel context prediction scheme that was developed in the scope of the thesis is also compared to the existing algorithms. For three exemplary algorithms, we provide simulation results in various application domains.

Zusammenfassung

Kontext Sensitivität beschreibt einen Forschungsbereich, in dem die Verbesserung der Interaktion zwischen Mensch und Gerät im Vordergrund steht. Der Fokus wird hierbei auf die Reaktion des Gerätes auf Reize aus der Umwelt gelegt. Durch dieses Vorgehen soll das Wissen von Anwendungen über die Bedürfnisse des Nutzers über explizite Nutzereingaben hinaus ausgedehnt werden. Die Motivation für dieses Vorgehen liefert die Erkenntnis, dass eine gegebene Situation und Umgebung weitreichenden Einfluss auf die Intention des Nutzers hat. Dies lässt sich am Beispiel des Klingelprofils von Mobiltelefonen verdeutlichen. In einer Besprechung wird ein Nutzer vermutlich ein stumm geschaltetes Klingelprofil bevorzugen, wohingegen er in der Zeit zwischen zwei Besprechungen durch einen Klingelton auf eingehende Anrufe aufmerksam gemacht werden möchte. Das bevorzugte Klingelprofil ändert sich in Abhängigkeit von der Situation in der sich der Nutzer befindet. Im Kontext-sensitiven Bereich wird die umgebende Situation mit Hilfe von Sensoren aufgezeichnet. Aus diesem Wissen wird zusätzliche Information extrahiert, die dazu beitragen soll, besser auf die Bedürfnisse eines Nutzers reagieren zu können.

Kontext-Sensitivität beschreibt ein weites Feld, das in weitere Disziplinen aufgeteilt werden kann. Für ein Kontext-sensitives Gerät ist es nicht implizit klar, ob das Gerät vergangene, gegenwärtige oder zukünftige Kontexte verarbeitet. Die meisten Arbeiten in diesem Bereich berücksichtigen die Verarbeitung gegenwärtiger oder vergangener Kontext-Informationen. Die Nutzung zukünftiger Kontext-Informationen ist dagegen weit weniger intensiv untersucht. Dieser Forschungszweig wird als Kontext-Prognose bezeichnet.

In der vorliegenden Arbeit betrachten wir Fragestellungen mit Bezug zu zukünftigen Kontexten sowie der Nutzung von vergangenen und gegenwärtigen Kontext-Informationen in diesem Zusammenhang. Wir untersuchen das Problem der Kontext-Prognose und identifizieren Chancen und Herausforderungen in diesem Bereich. Diese Diskussion führt zu einer Definition des Kontext-Prognose Prozesses.

Unterschiedliche Kontext-Prognose-Verfahren können unterschieden werden. Abhängig von der Menge an Kontext-Vorverarbeitung, beziehungsweise abhängig von dem Kontext-Abstraktionsgrad unterscheiden wir zwischen verschiedenen Kontext-Prognose-Verfahren. Wir untersuchen Kontext-Prognose-Verfahren bezüglich ihrer Genauigkeit und stellen Richtlinien für Anwendungs-Entwickler bereit, welches Kontext-Prognose-Verfahren in einer gegebenen Situation zur Anwendung kommen sollte. Analytische Ergebnisse aus dieser Betrachtung werden durch Simulationsergebnisse in verschiedenen Anwendungsszenarien bestätigt.

Um Kontext-Prognose-Verfahren eine breite Anwendung in verschiedenen Anwendungsszenarien zu ermöglichen, können standardisierte Kontext-Prognose-Architekturen hilfreich sein. Derzeit existieren nur wenige Ansätze für derartige Architekturen. Diese Ansätze sind, bezogen auf den Kontext-Abstraktionsgrad, als statisch zu bezeichnen. Wir entwickeln eine Architektur für Kontext-Prognose, die bei beliebigem Kontext-Abstraktionsgrad Verwendung finden kann. Diese Architektur lässt zudem eine verteilte Implementierung auf verschiedenen, untereinander kommunizierenden Geräten in mobilen, ubiquitären Szenarien zu.

Ein weiterer Aspekt von Kontext-Prognose-Verfahren sind Algorithmen für dieses Vorgehen. Wir stellen gegenwärtig verwendete Ansätze vor und untersuchen Stärken und Schwächen der einzelnen Verfahren. Anhand identifizierter Anforderungen für diese Algorithmen in Kontext-Prognose-Szenarien finden wir Ansätze, die am besten für Kontext-Prognose-Verfahren geeignet sind. Ein von uns entwickelter, neuartiger Kontext-Prognose-Ansatz wird zudem mit bestehenden Algorithmen verglichen. Für drei exemplarische Algorithmen stellen wir Simulationsergebnisse aus verschiedenen Anwendungsszenarien vor.

Acknowledgements

Various people have contributed to this thesis in several ways. My work in conjunction with this thesis has profited much from these positive contributions. First of all, I would like to thank Professor Dr. Klaus David for his constant and intriguing questioning of the results obtained and for the ensuring and definitely fruitful discussions. This work has greatly benefited from these stimulating exchanges. Also, many thanks to Professor Dr. Alois Ferscha. Many thoughts from his publications have motivated me and also influenced the current document. I am grateful that he accepted my request to be my second adviser in this thesis. I also much appreciate that Professor Dr. Albert Zündorf as well as Professor Dr. Kurt Geihs accepted to review this thesis.

I would further like to thank the team at ComTec for absorbing discussions on context-aware topics and for a great time at the University of Kassel. In particular, I would like to thank Dr. Sandra Haseloff for her advice and guidance during most of the writing process and for bringing me back on track when I was about to lose touch. A really great job has been done by Tom Norberg. Thank you for your meticulous proof-reading and for helping me improve my written English.

I also thank my colleague Tino Löffler, for stirring discussions on context-aware topics. As a result of these discussions, I was challenged to re-think several approaches and ended up with even better conclusions. I am much obliged as well to my former colleague Dr. Guihua Piao. Although we had quite different research topics, her endurance and consistency impressed me and later inspired my work.

A great help through implementation of algorithms and demonstrators as well as through intriguing discussions on these topics was provided by Alexander Flach, Christof Hoppe and Elmir Rustemovic. Thank you for contributing to improve the algorithmic part of this work as well as the simulation results.

Furthermore, I am very grateful to my parents for supporting me and to my family, Nelja and Freyja. Thank you for your patience and under-

standing when I spent nearly all my time at work and for cheering me up when necessary.

Особенную благодарность хочу выразить моей семье, которая, несмотря на невозможность прочтения кандидатской, проявила терпение и поддержку.

Finally, I would like to acknowledge partial funding by the German "Bundesministerium für Bildung und Forschung (BMBF)" in the framework of the Wireless Internet and mik21 projects.

Contents

Abbreviations and Notation

The following figures and notations are utilised throughout this thesis. It has been attempted to keep the standard notation from the literature whenever possible. However, since the thesis touches diverse scientific areas, the notation had to be adapted for this thesis in order to provide an unambiguous notation. The page number given in the table refers to the first occurrence of the mentioned construct.

Notation	Explanation	Page
AP	Alignment prediction algorithm	231
AR	Autoregressive	227
ARMA	Autoregressive Moving average	79
BN	Bayesian Net	209
C	A set of context elements	214
c_i	A context element	47
χ_i	Observation or event of a stochastic process	79
d_i	Actual value of the context element at time t_{0+i}	81
δ	Error threshold of one context source	157
DIM	Time series dimension	176
$dim(T)$	The dimension of time series T	71
ETH	Swiss Federal Institute of Technology Zürich, Switzerland	40
f	Numerical functions are denoted with f	72
$G = (V, E)$	A graph G with vertices in V and edges in E	67
GPS	Global Positioning System	18
GSM	Global System for Mobile Communications	17

Notation	Explanation	Page
ID	Identification	46
ISET	Institut für solare Energieversorgungstechnik, Kassel, Germany	63
IST	Priority of the 6th framework programme of the European Union: Information Society Technology	42
k	Context history size	75
KM	Kernel Machines	209
λ	An empty time series.	70
LEN	Context history size	176
M	Matrices are denoted with M	219
m	Number of context sources and low-level contexts in one time interval. Dimension of low-level context time series.	123
MEG	Mobile Event Guide	112
MIT	Massachusetts Institute of Technology	37
MM	Markov Models	209
n	Length of the prediction horizon	81
NN	Neural Nets	209
NNS	Nearest Neighbour Search	209
o	Number of context high-level contexts in one time interval. Dimension of high-level context time series.	123
P_{acq}	Probability that no error occurs in the context acquisition process	122
P_{hl}	Probability that context prediction based on high-level context elements is without error	125
p_i	Outcome of the prediction process of the context element at time t_{0+i}	81
P_{int}	Probability that no error occurs in the context interpretation process	122

Notation	Explanation	Page
P_{ll}	Probability that context prediction based on low-level context elements is without error	127
$P_{ll}(i), P_{hl}(i)$	Probability that the prediction based of the i-th context element is without error for low-level and high-level context prediction schemes respectively.	125, 126
P_{pre}	Probability that no error occurs in the context prediction process	122
π	A stochastic process	80
PM	Pattern Matching	209
PyS60	Python for Symbian Series 60 platform	114
RMSE	Root of the Mean Squared Error	238
S	A search space	156
S_l, S_h	Search spaces of high-level and low-level context prediction schemes	156
S60	Symbian Series 60	114
SOM	Self Organising Map	96
SPM	State Predictor Method	231
SVM	Support Vector Machine	209
T, T'	Time series	70
T_{t_j,t_k}	Time series T in the interval $[t_j, t_k]$	70
$\|T\|$	Number of time series elements in time series T	71
t_i	A time interval	47
τ	A learning threshold	197
TecO	Telecooperation Office Karlsruhe, Germany	39
noalign TS	Time series	181
UbiComp	Ubiquitous Computing	45
UMTS	Universal Mobile Telecommunications System	17
v_i	Measured context value at time t_{0+i}	158
$\overrightarrow{v}$	A vector $v = (v_1, \ldots, v_\kappa)$	210

Notation	Explanation	Page
v_l	Number of legal values for low-level context time series elements	123
v_h	Number of legal values for high-level context time series elements	123
WLAN	Wireless Local Area Network	17
ξ_i	Time series element	71

1 Introduction

> *History has shown that forecasting the future has become a science and perhaps even an art.*
>
> (P. Duin and R. Kok, Mind the gap - linking forecasting with decisionmaking. [1])

The vision of context-awareness is that applications become sensitive to environmental stimuli and adapt their behaviour to the current situation. This vision was far ahead of the technology of the time when it was first studied in research laboratories and the details necessary to implement the vision were seldom provided. With improved technology we have seen prototype applications of isolated ideas from the Context-aware vision become implemented. The first of these are probably the Xerox PARCTAB [2] and the media cup [3].

In recent years, but to a limited degree, we have already seen context-aware features in consumer products. Mobile devices that adjust their screen brightness to the environmental light, devices that automatically rotate the screen when the device is turned, watches that automatically adjust to local time and messages that alert users when their screen work time exceeds a certain limit, are just some examples.

While these applications are quite limited and stand alone, we see more advanced and better integrated context-aware features in multifarious new products. The most versatile and widely used device type for context-aware applications are recent mobile phones. The capabilities of these devices quickly increase as new interfaces to the environment are constantly added. Apart from technologies as basic as microphones, speakers and GSM, we now expect also infrared, bluetooth and a camera in mobile devices. New air interfaces as WLAN or UMTS are added, as well as light

sensors, accelerators, touch screens and to an increasing degree GPS receivers. Most of these technologies remain unused for a great part of the time. This multitude of sensors, however, provides a rich environment in which context-aware applications can be taken to the next evolutionary stage. Context-awareness, nowadays, still holds great potential before the development comes anywhere close to the vision of a ubiquitous world that is saturated with context-aware devices.

Some branches of context-awareness have still not left the research laboratories. A topic that, until now, holds lots of open research questions is context prediction. The idea of context prediction is basically to expand the awareness of an application on observed contexts into the future. Applications that become possible with context prediction are numerous. A few examples shall illustrate the enormous potential of context prediction.

A context prediction capable application can, for instance, foresee interdependencies that are hard to keep track of for a user due to their high complexity. Consider, for example, a device that automatically informs the person you are about to meet that you will be delayed by a traffic jam or due to a delayed train even before you are actually late. Furthermore, in mobile scenarios, prediction of resource consumption of mobile users might contribute to the improvement of the overall network capacity. Also, if the availability of an individual in her office is predicted for potential visitors, these could more efficiently schedule their appointments with the person in question.

A broad spectrum of alternative application scenarios for context prediction approaches is presented in [4, 5]. Recently, an initial study on context prediction has been conducted in [6]. One main focus in this work is on an architecture for context prediction. Basically, decent ideas applied for context-aware architectures are enhanced by a context prediction layer. Hence, the architecture contains, apart from context prediction features, mechanisms to acquire contexts from sensors. Consequently, context clustering and context prediction mechanisms have been studied. However, various open questions remain for context prediction.

Key challenges for context prediction are discussed in section 1.1. Only after these challenges have been addressed will applications for context prediction scenarios become feasible. Challenges addressed in the course of this thesis are described in section 1.2. The basic approach taken in order to solve the problems at hand is introduced in section 1.3. Section 1.4 summarises the contribution of this thesis.

1.1 Challenges

Challenges for context prediction approaches start with matters as essential as needing a formal definition of the context prediction task. Several authors who have approached context prediction tasks provide informal descriptions of the problem their algorithms are solving, or provide an abstract idea of the desired output of the prediction process (Compare, for example, [4, 6, 7]). However, no formal description of the context prediction task that contains a description of the input and output data, as well as the prediction aim of the prediction process, has been provided so far.

Another challenge is posed by the prediction architecture. Few approaches to this question have yet been published [4, 6]. However, these are too restricted, focusing on specific classes of applications, to be generally applicable for context prediction tasks. In particular, the application to contexts of arbitrary context abstraction levels is not considered by these architectures.

Furthermore, for context prediction algorithms, a thorough overview and comparison has not been provided until now. The author of [6] provides a first comparison of selected algorithms but leaves out others. Additionally, the application domains these algorithms are applied to are few. Apart from the algorithms that are not considered, other, new context prediction algorithms have been proposed in [7] and [8]. A comprehensive discussion on these context prediction algorithms is still lacking.

Requirements for context prediction algorithms have also not yet been stated comprehensively. Requirements that are commonly referred to are high prediction accuracy as well as low processing cost along with low memory consumption. However, also other properties might prove useful for context prediction approaches. Candidates are, for example, a high adaptability to changing environments, as well as applicability to contexts of arbitrary context data type. Various algorithms have to be compared in identical application domains and for varying configuration parameters. Interesting questions raised by this challenge are the relation between the length of the prediction horizon and the prediction accuracy, as well as on the influence of the memory consumption and runtime related to the accuracy.

Probably the most serious challenge for context prediction tasks is related to the prediction accuracy. The usefulness of context prediction approaches is directly connected to the quality of the prediction in terms

of accuracy. Although accuracies for distinct algorithms with unique configurations that are applied to typical application domains are known, a more general discussion is missing.

Further aspects that are independent of the prediction algorithm and that impact the prediction accuracy are not yet considered. Likely influences on the prediction accuracy originate, for example, from the number and type of sensors utilised for prediction tasks as well as from the context abstraction level.

1.2 Problem statement

This thesis contends that, up to now, the task of context prediction in ubiquitous computing environments and especially impacts on the context prediction accuracy have yet not been covered comprehensively. Furthermore, algorithms utilised for context prediction in ubiquitous computing environments so far do not utilise all domain specific information available. Finally, a straight and agreed upon definition, as well as standards for the context prediction task, have not been worked out extensively.

These three issues constitute the main research emphasis of this thesis. Considering the latter issue, we focus on two basic domains. The one is a definition of the context prediction task. The other is an architecture for context prediction that can be widely applied to arbitrary context prediction scenarios.

The architectures for context prediction that have so far been developed are typically multi-layered and restricted to specific cases of context prediction. Since the order of essential processing operations on contexts is predefined by these architectures, the context abstraction level of the context prediction process is to a large extent fixed. We develop a one-layered architecture for context prediction that is flexible in the sense that it may be distributed among devices, as well as being able to be applied to contexts of arbitrary context abstraction level.

Another focus of this document is on the context prediction accuracy. The accuracy of the context prediction task is a most serious issue that impacts the usability of context prediction applications. We study three basic aspects that affect the context prediction accuracy.

A further aspect we consider is the impact of the context abstraction level on the context prediction accuracy. Higher abstraction levels typically relate to additional processing of contexts. Since every processing

operation might induce errors into the context data, higher context abstraction levels might also incorporate a higher error count even before the context prediction is applied. We study the impact of the amount of pre-processing on the context prediction accuracy.

The third aspect considered covers the impact of the size of the search space on the context prediction accuracy. Due to a differing context abstraction level, the search space of context prediction algorithms might differ. Search spaces of lower abstraction levels are typically more expressive and contain more elements than search spaces of higher abstraction levels. We study the impact of the search space dimension on the context prediction accuracy.

Finally, a learning approach implemented in conjunction with context prediction might be affected by the context abstraction level. Since the error count for higher abstraction levels is higher, learning might be impaired. Learning can be seen as keeping those patterns in memory that are considered typical. Consequently, when incorrect patterns are learnt, this also affects the long-term accuracy of a context prediction algorithm. We study the long-term accuracy of context prediction approaches of varying context abstraction levels.

Another issue we consider that also impacts the context prediction accuracy regards the context prediction algorithm utilised. Algorithms that are better suited for a given context prediction scenario are better capable of providing a high prediction accuracy. A comprehensive study of a vast number of context prediction algorithms for their suitability to context prediction tasks is still not available. We develop requirements for context prediction algorithms and study common algorithms that are utilised for context prediction according to the suitability to fulfil these requirements. A set of best suited algorithms for context prediction tasks is identified.

1.3 Scope and methodology

For the three main issues context prediction accuracy, standards for context prediction and context prediction algorithms, the methodology applied differs slightly. However, in all cases we take a computation-centric approach. This means that we are considering general aspects that are inherent to the computation of arbitrary context prediction algorithms in arbitrary context prediction scenarios rather than aspects for specific scenarios or algorithms.

In case of an architecture for context prediction and a definition for the context prediction task, the environment for context prediction in ubiquitous computing scenarios is first analysed in order to find requirements that are necessary in order to fit the given task.

Following these requirements, which can also be seen as boundaries or restrictions, a most suitable design of the context prediction architecture, as well as a most suitable definition for context prediction, is found.

For the study of algorithms suited for context prediction, we initially review requirements for context prediction algorithms and afterwards study typical context prediction algorithms for their ability to fulfil these requirements. For those algorithms that are identified as best suited for context prediction tasks, we conduct simulations in two distinct context prediction scenarios in order to evaluate the performance of the algorithms.

Regarding the aspects that impact the context prediction accuracy, we take a different approach. In order to estimate the impact of the amount of pre-processing applied on context data, the dimension of the search space for context prediction as well as the long-term accuracy, we first provide an analytic consideration of the problem scenario in all three cases respectively. Simulations are conducted that confirm the analytic findings. These simulations are executed on realistic data samples as well as on synthetic data. In the synthetic simulations we isolate discrete aspects of the scenario that are otherwise fused together in realistic simulations.

1.4 Contribution

This thesis provides a formal definition of the context prediction problem. This is the first definition of its kind and consequently closes one gap to a comprehensive understanding of the context prediction task.

In addition, a flexible and lean architecture for context prediction is developed. This architecture differs from other architectures for context prediction mainly in the scope of context processing operations that are included. Due to a reduced scope on context prediction related tasks exclusively, a more general approach is possible. The proposed architecture is applicable to context data of arbitrary context abstraction level. Additionally, the architecture supports a distribution of components among various devices and allows for a non-continuous availability of components at execution time.

We propose the alignment prediction approach to be utilised for context

prediction tasks. The alignment prediction approach is a specialised context prediction algorithm that was developed in the course of this thesis. It implements an approximate pattern matching mechanism and is especially well-suited to distinguish typical context patterns in a sequence of contexts.

Moreover, we recapitulate typical algorithms that are utilised for context prediction tasks and summarise their strengths and weaknesses. A comparison of these algorithms regarding requirements stated for context prediction approaches is provided. We discuss the suitability of the alignment prediction algorithm along with the suitability of the other algorithms mentioned, to context prediction tasks and compare the alignment prediction algorithm to popular prediction approaches in simulations on real data.

We also provide extensive studies on impacts on the context prediction accuracy. The impact of context processing on the context prediction accuracy is studied analytically and in simulations on real and synthetic data sets. For the analytic consideration, a uniform distribution of errors inferred by context processing is assumed, as well as further aspects of the prediction environment. Guidelines to adapt the calculations for specific scenarios are provided.

Apart from the impact of context pre-processing, the influence of the long-term accuracy is studied analytically. Additionally, simulations on real and synthetic data sets are conducted in order to confirm the properties observed in the analytical discussion. Again, for the analytic consideration the general case is considered rather than a special scenario or algorithm.

Regarding the influence of the search space dimension, the general problem is motivated and impacts are demonstrated in simulation scenarios on real and synthetic data sets.

1.5 Outline of the thesis

This document is organised as follows. In chapter 2, the research field of context-awareness is introduced. Section 2.1 recapitulates recent work as well as introduces related research groups and projects whereas section 2.2 discusses concepts and definitions that are relevant for this thesis. One of the concepts introduced is the notion of context abstraction levels.

Chapter 3 covers general context prediction related topics. In section 3.1, related projects and research groups are introduced. Section 3.2 discusses concepts and definitions for context prediction that are utilised throughout this thesis. A definition for the context prediction task is provided in section 3.2.4.

Chapter 4 introduces a modular architecture for context prediction. After a first discussion of general requirements for context prediction architectures in section 4.1, the distinct modules and implementation details for the proposed architecture for context prediction are introduced in section 4.2. Section 4.3 describes an application scenario the architecture was applied to.

In chapter 5, several aspects that impact the context prediction accuracy are discussed. Section 5.1 discusses impacts of the context abstraction level on the context prediction accuracy. In section 5.1.1 the impact of the context pre-processing is studied, while section 5.1.2 discusses impacts on the context prediction accuracy due to the size of the prediction search space. In section 5.2 we study the long-term accuracy of context prediction for various context abstraction levels.

Chapter 6 introduces various context prediction algorithms and requirements for context prediction tasks. Section 6.1 discusses requirements for context prediction algorithms, whereas in section 6.2 various typical prediction approaches are studied for their suitability to comply with the requirements stated. Implementation related issues for the set of algorithms that are identified as best suited in context prediction scenarios, are discussed in section 6.3. Section 6.4 provides simulation results of these algorithms in two distinct context prediction scenarios.

In chapter 7 conclusions are presented.

1.6 Publications

Parts of the work conducted for this thesis has already been published at conferences or workshops. These publications are

- Stephan sigg, Sandra Haseloff, Klaus David: A Study on Context Prediction and Adaptivity. In: Proceedings of the International Workshop on Context Modeling and Management for Smart Environments (CMMSE'07), Oktober 28-31, Lyon, France. 2007

- Stephan Sigg, Sandra Haseloff, Klaus David: Prediction of Context

Time Series. In: Proceedings of the 5th Workshop on Applications of Wireless Communications (WAWC'07), August 16, Lappeenranta, Finland. 2007

- Stephan Sigg, Sian Lun Lau, Sandra Haseloff, Klaus David: Approaching a Definition of Context Prediction. In: Proceedings of the Third Workshop on Context Awareness for Proactive Systems (CAPS 2007), June 18-19, Guildford, United Kingdom. 2007

- Stephan Sigg, Sandra Haseloff, Klaus David: Minimising the Context Prediction Error. In: Proceedings of IEEE 65th Vehicular Technology Conference VTC2007-Spring, April 22-25, Dublin, Ireland. 2007

- Stephan Sigg, Sandra Haseloff, Klaus David: A Novel Approach to Context Prediction in Ubicomp Environments. In: Proceedings of the 17th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2006), September 11-14, Helsinki, Finland. 2006

- Stephan Sigg, Sandra Haseloff, Klaus David: The Impact of the Context Interpretation Error on the Context Prediction Accuracy. In: Proceedings of the 3rd Annual International Conference on Mobile and Ubiquitous Systems: Networks and Services (MOBIQUITOUS 2006), July 17-21, San Jose, CA. 2006

- Stephan Sigg, Sandra Haseloff, Klaus David: Context Prediction by Alignment Methods. In: Poster Proceedings of the 4th International Conference on Mobile Systems, Applications, and Services (MobiSys 2006), June 19-22, Uppsala, Sweden. 2006

- Tino Loeffler, Stephan Sigg, Sandra Haseloff, Klaus David: The Quick Step to Foxtrot. In: K. David, O. Droegehorn, S. Haseloff (eds.): Proceedings of the Second Workshop on Context Awareness for Proactive Systems (CAPS 2006), June 12-13, Kassel, Germany. Kassel University press, 2006

2 Context-awareness

> *Increasingly, the bottleneck in computing is not its disk capacity, processor speed or communication bandwidth, but rather the limited resource of human attention*
>
> (A. Garlan, Toward Distraction-Free Pervasive Computing [9])

In recent years, applications and devices have undergone serious changes that move them away from static, reactive entities towards a more environment responsive design. We see applications act in an increasingly adaptive and situation-dependent way. Applications are able to infer the needs and requirements in a given situation. It is commonly agreed that the general setting a user is in also influences her needs at that point in time. Lucy Suchman [10] states that every course of action is highly dependent upon its material and social circumstances regarding interactions between actors and the environment. To become able to react to the general setting an application is executed in, the design paradigm for applications is shifting from an application-centric approach to an environment-centric approach. Applications become integrated into the environment and react to environmental stimuli. In order to improve the application and device behaviour in this direction, further and in most cases novel sources of information are investigated.

The input provided to an application or device is no longer restricted to explicit instructions on a common user interface. Instead, the interface utilised for the acquisition of input information is extended and coupled by an interface to the environment. The behaviour of applications evolves from a mere passive, input dependent way to an active, environment and

situation guided operation.

Information about the environment and situation is extracted and interpreted to trigger situation dependent actions that shall for example provide the user with a richer experience that is adapted to her personal needs. Due to this additional information, the required explicit interaction with an application can be minimised or at least reduced. The computing experience hereby gets increasingly unobtrusive and becomes ubiquitous.

In general, this computing paradigm is referred to as context-awareness or context computing but is described by various further titles. People have been quite creative in finding descriptive names for scenarios similar to the one described above. A (most certainly not exhaustive) set of terms associated with ideas related to context computing is depicted in figure 2.1. A similar list can also be found in [11]

While these catchwords have partly redundant but not identical meanings, a common vision of future computing is captured by all these descriptions. Probably the first study on context-aware computing was the Olivetti Active Badge [12]. Following this pioneering work, numerous further concepts and ideas have been discussed by various research groups. We discuss this development in section 2.1. In section 2.2 we introduce several concepts related to context-awareness that are required for the remainder of this thesis. Section 2.3 summarises our discussion in this section.

2.1 Context-aware computing

The vision of a world where computing devices seamlessly integrate into the real world was first introduced by Mark Weiser in 1988. He illustrates and describes his vision of future computing in [13]. Computing in his vision is no longer restricted to a single machine but may move off one machine and onto another one at execution time. Ubiquitous computing also incorporates an awareness of the environment the computer is situated in. Furthermore, following the vision of ubiquitous computing, computing becomes invisible and omnipresent simultaneously. Smallest scale computing devices that enrich the environment communicate with each other and assist a user unnoticed. Weiser argues that a computer might adapt its behaviour in a significant way if it knows where it is located. As Weiser states, this reaction to the environment does not require artificial intelligence.

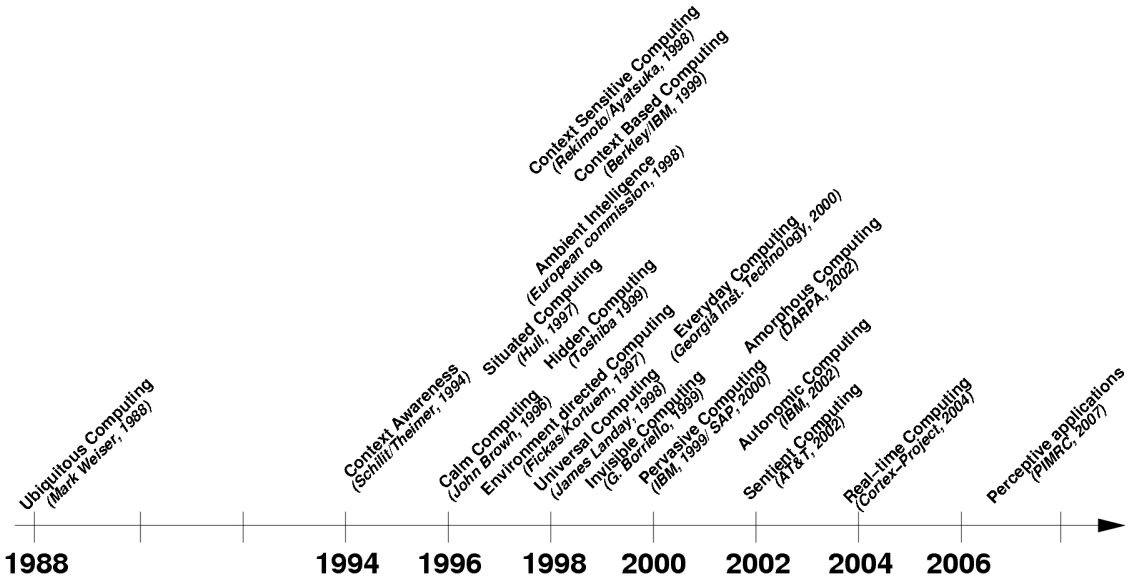


Figure 2.1: Concepts related to Ubiquitous computing

Weiser observes the paradox that computing devices are becoming cheaper, smaller and more powerful at the same time. Tiny computing devices become cheap enough to be bought in raw amounts and small enough to be integrated in virtually every real world object.

Weiser envisions that these devices, equipped with sensing technology and communication interfaces are able to communicate with each other and to acquire and spread information on devices, persons and objects in their proximity. This information can then be utilised to enhance the computing experience of a user.

The first experiments with computers aware of their environment have been conducted in the early 1990's. The active badge location system by Olivetti Research [12] and the Xerox PARCTAB location system by Xerox laboratories [2] demonstrated how small mobile devices operate together.

Although the sources of information utilised in these experiments were restricted to location sensors, the basic new concept and possibility inspired numerous people to focus their research on this field.

2.1.1 Definitions of context

Definitions of context are numerous and diverse even when the focus is restricted to computer sciences. In his comprehensive discussion "What we talk about when we talk about context"[14] Paul Dourish attempts to exhaustively discuss several aspects of context and also reviews various definitions of context.

The concept of context in conjunction with context-aware computing was first formulated by Schilit and Theimer in 1994 [15]. Following their definition, a software that "adapts according to its location of use, the collection of nearby people and objects as well as changes to those objects over time" is considered to be context-aware. Later on, Schilit refined this definition by defining context categories in [16]. These categories are 'user context', 'physical context' and 'computing context'. As further categories, Brown added information about the time [17], while Pascoe also considered the blood pressure of users [18]. Dey took the latter proposal to a broader scope by considering emotions and the focus of attention [19].

At about the same time, Albrecht Schmidt, Michael Beigl and Hans W. Gellersen recognised that most so-called context-aware applications are in fact location-aware [20]. Hence, they are considering only location as an aspect of the context. The assertion of the authors is that applications

implemented on mobile devices might significantly benefit from a wider understanding of context. Furthermore, they introduce a working model for context and discuss mechanisms to acquire other aspects of context beside location.

In their working model for context, they propose that a context describes a situation and the environment a device or user is located in. They state that a context shall have a set of relevant aspects to which they refer as features.

These features are ordered hierarchically. At the top level a distinction between human factors and physical environment is made. Further, finer grained sub-division of these top-level categories are also proposed. Finally, an overview of available sensor types and contexts obtained from these sensors is given.

As a prerequisite to a definition of context-awareness, Anind K. Dey formulated a definition of context, that is most commonly used today [21].

Definition 2.1.1 : User context

Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves.

This definition, while useful, is quite abstract and gives no hint on the actual representation of context in a computing system. For this reason, several authors express criticism considering this definition. As Jani Mäntyjärvi has already stated in [22], this context definition does not result in a more exact definition of context since the abstraction is shifted from context to information.

Karen Henricksen follows the same line of argumentation by remarking that the definition remains too imprecise, since a clear separation of the concepts of context, context modelling and context information is not provided. Henricksen refines the definition of context given by Dey as the set of circumstances surrounding a task that are potentially relevant for its completion [23]. Furthermore, in the model of Henricksen, a context model identifies a subset of the context that is realistically attainable from sensors, applications and users. Following her discussion, context information describes a set of data that was gathered from sensors and users and that conforms to a context model.

However, the discussion about a most suitable definition is not settled yet. In 2000, Lieberman and Selker defined context to be any input other

than the explicit input and output [24]. Other projects refine the definition of context to their individual needs. In [25] for example, the definition of Dey is refined by adding the concept of a sentient object.

A discussion on the definition of context we utilise in our work is given in section 2.2.3.

2.1.2 Context-awareness

Intuitively, applications that utilise context data are context-aware. However, similar to the lively discussion on a definition of context, several definitions for context-awareness have been given in the literature. This section briefly reviews this ongoing discussion.

In [15] Schilit and Theimer formulated a first definition of context-awareness. Following this definition, "Applications are context-aware when they adapt themselves to context".

In 1998 Pascoe argues that context-aware computing is the ability of devices to detect, sense, interpret and respond to changes in the user's environment and computing devices themselves [26]. The authors of [27] define context-awareness as the automation of a software system based on knowledge of the user's context. Several other similar definitions treat it as applications' ability to adapt or change their operation dynamically according to the state of the application and the user [15, 17, 28].

Later, Dey argued that the existing definitions did not fit to various applications developed at that time that were intended to be context-aware and consequently stated a more general definition of context-aware systems in [21].

Definition 2.1.2 : Context-awareness

A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.

This discussion is not closed yet as several research groups refine the definition so that it best suits their needs (cf. [25]).

2.1.3 Context processing

Context is an abstract concept to describe a major input of ubiquitous computing applications. However, we cannot build applications with this theoretical construct. The questions are how context can be obtained from

the available information sources, in which way context is represented in applications and how context can be further processed. This section discusses popular approaches to these questions.

Various authors propose to pre-process sensor output in order to prepare the data for further computation. Anind K. Dey argues that one of the main reasons why context is not used in applications is because no common way to acquire and handle context is specified [21]. He proposes to separate the context acquisition from the context utilisation process. Dey distinguishes between two basic forms of context. Raw or low-level context data that is directly acquired by sensors and richer or higher-level forms of information. A similar distinction is also made by Guanling Chen [29]. However, no concrete specification of these notions is given.

Albrecht Schmidt on the other hand argues that it is simpler to implement context-aware systems using contexts on entity level [30]. With the notion 'entity level', Schmidt refers to context data that is not further processed or aggregated after it has been obtained from context sources. Furthermore, intrinsic properties of sensors are utilised in the context modelling process. Schmidt refers to this approach as the concept of bottom-up context-awareness. The main research focus of Schmidt is related to context acquisition from a variety of simple sensors. He defines simple sensors as low-end, low-price computing and communication technology.

These ideas are utilised by Johan Himberg. Himberg studies data mining and visualisation for context-awareness and personalisation [31]. He especially focuses on sensor data captured by on-board sensors of mobile phones. He investigates how to infer context from features derived from the sensor signals. Johan Himberg especially only utilises simple statistical methods in order to reach his aim.

An approach focused on the whole process of context inference is proposed by Jani Mäntyjärvi. Mäntyjärvi considers the problem, how low-level contexts can be obtained from raw sensor data [22]. This problem is basically related to the extraction of features from information sources. For each context a set of features is relevant that determines the context. After the feature inference process, Mäntyjärvi composes the sampled features to obtain a more expressive description of a context. This operation is considered as the processing of low-level contexts to obtain high-level contexts.

Mäntyjärvi presents a procedure for sensor-based context recognition. This approach is referred to by him as bottom-up approach, in contrast

to a top-down approach that starts from the high-level context as it had been proposed by Dey in [21]. Included in this procedure is also a method to extract information on contexts and to convert it into a context representation. Following his definition, raw sensor data is sensor data like $24°C$, or 70% humidity. Low-level contexts are defined as pre-processed raw sensor data where the pre-processing may be constituted, for example, from noise removal, data calibration and reforming of data distributions. Generally, low-level contexts are conditions like 'warm' or 'normal humidity'. Higher level contexts are then created by an additional processing of low-level contexts that results in an action like 'having lunch'.

Main assumptions prior to his work are that sensors attached to computing devices have to be carefully chosen in order to be useful and that context actually can be recognised by sensor data.

The term context atom was introduced in [32] and has been used by Jani Mäntyjärvi, Johan Himberg and Pertti Huuskonen in to describe basic context dimensions which are derived from low-level sensor data by pre-processing [33].

2.1.4 Frameworks and architectures for context-awareness

In order to facilitate the development of context-aware applications, several authors have proposed frameworks and architectures for this task.

In his PhD thesis in 1994 [34], Schilit concludes that traditional software approaches are not well-suited to building distributed mobile systems. The main reason for this dilemma is that applications are seldom designed to adapt their behaviour to the ever-changing mobile environment of the user in which they are executed. By designing an architecture that communicates context changes to the application, Schilit proposes a solution to this problem.

Additionally, Schilit identifies the problem that the user context may not be shared by distinct applications, although they are actually executed in the same user context. Schilit proposes the use of a user agent that administers the user context in order to provide a persistent dynamic context for all applications of the user.

Furthermore, he presents a system structure for use with context-aware systems. He recommends a distribution of system functions and designs protocols for communication between the entities.

These thoughts are further developed in the context toolkit that was

introduced in 2000 [21]. It was proposed and developed by Anind K. Dey at the Georgia Institute of Technology. The context toolkit constitutes a conceptual framework that was designed to support the development of context-aware applications. It is widely accepted as a major reference for context-aware computing. An important contribution of this framework is that it distinguishes between context sensing and context computing. Context sensing describes the process of acquiring information on contexts from sensors while context computing refers to the utilisation of acquired contexts. Basic components in this architecture are context widgets (encapsulated sensors), aggregators and interpreters. However, the Context Toolkit is not generally applicable for arbitrary context-aware applications since it exclusively features discrete contexts and does not consider unreliable or unavailable sensor information [35].

Later on, Albrecht Schmidt presented a "working model for context-aware mobile computing" which is basically an extensible tree structure [30]. The proposed hierarchy of features starts with distinguishing human factors and the physical environment and expands from there. One of the major contributions of his PhD thesis is a framework supporting design, simulation, implementation and maintenance of context acquisition systems in a distributed ubiquitous computing environment.

In 2003, Karen Henricksen introduced a novel characterisation of context data in ubiquitous computing environments [23]. Her introductory study of the ubiquitous computing environment especially focuses on challenges in providing computing applications in ubiquitous computing environments. These issues can be summarised as the autonomy of computing applications, dynamic computing environments, dynamic user requirements, scalability and resource limitations. Henricksen concludes that this set of challenges necessitates a new application design approach. Henricksen proposes a conceptual framework and a corresponding software architecture for context-aware application development.

This framework consists of programming models to be used for context-aware systems. Furthermore, Henricksen proposes the use of the Context Modelling Language (CML), a graphical notation of context that supports the specification of application requirements by the application designer.

In 2004 the Solar framework was presented by Chen [29]. It provides means to derive higher-level context from lower level sensor data.

The framework basically represents a network of nodes that interact with each other. It is scalable, supports mobility of nodes and is self

managed.

Solar is designed as a service-oriented middleware in order to support the distribution of its components. The middleware supports sensors, as well as applications. Components and functions can be shared between applications. The data flow between sensors and applications may be composed as a multi-layered acyclic directed graph both at design time or at runtime.

Together with Solar, Chen provides a graph-based programming model, that can be utilised for the design of context-aware architectures.

2.1.5 Applications utilising context

Several applications that utilise context have been developed in recent years. In this section we introduce a set of applications that illustrate the uses and application fields of context-aware computing applications. The number of context-aware applications has reached an immense quantity. It is beyond the scope of this document to present an exhaustive overview of these applications. The examples presented are chosen in order to illustrate the broad spectrum of approaches and to show the possibilities for context-aware applications.

With the MediaCup [3], Hans W. Gellersen, Michael Beigl and Holger Krall have presented a context-aware device that demonstrates one part of Mark Weiser's vision of ubiquitous computing. The MediaCup is a coffee cup that is enriched with sensing, processing and communication capabilities. The cup was developed to demonstrate how ordinary, everyday objects can be integrated into a ubiquitous computing environment. The context data obtained by the cup is related to the location of the cup, the temperature and some movement characteristics. This information is obtained by a temperature sensor and an acceleration sensor. Context information can be broadcast with the help of an infrared diode. The MediaCup has been utilised in research projects in order to provide a sense of a remote presence and in order to log user activity.

Another application proposed by Gellersen et al. is context acquisition based on load sensing [36]. With the help of pressure sensors in the floor of a room, the presence and location of objects and individuals can be tracked. Furthermore, it is shown that it is possible to distinguish between objects and that even movement of objects can be traced. The authors consider the use of load sensing in everyday environments as an approach

to acquisition of contextual information in ubiquitous computing systems. It is demonstrated that load sensing is a practical source of contexts. It exemplifies how the position of objects and interaction events on a given surface can be sensed.

Various implemented context-aware applications have been developed by the Context-Aware Computing Group at the MIT[1]. An illustrative example is the 'Augmented Reality Kitchen' that monitors the state of objects in a kitchen in order to help the kitchen-worker to keep track of all simultaneous events. The kitchen displays the location of tools and the state of cooking processes. In the related project 'KitchenSense', a sensor-rich networked kitchen is considered that attempts to interpret peoples' intentions and reacts accordingly.

Additionally, the SenseBoard has been proposed in [37]. The SenseBoard approach is to combine the benefits of the digital world with those of the real world. The SenseBoard is a hardware board with a schedule projected onto it. Discrete information pieces that are stored in a computer can be manipulated by arranging small items on the board. These items are entries of the schedule. The naming of each item is computer-controlled and projected onto the item. Like in a digital schedule, items can be easily arranged, grouped together or expanded. Operations and the status of the schedule are projected to the physical schedule on the board. Like with real-world objects, people can manually arrange the items on the hardware board. This makes the operation more intuitive and enables the participation of larger groups in the process of finding an optimal schedule for a given task. Detailed information on each item can be made available and a schedule can be digitally exported, stored or loaded and also printed.

2.1.6 Context-awareness research groups and projects

Several research groups at various Universities and from industry are considering issues in context-aware computing. Of course, this chapter does not claim to cover the overwhelming quantity of institutes that consider context-aware computing entirely. Alternatively, and in order to provide a more structured overview of work conducted in this field, in this section we review some projects and institutes that examine questions that are most closely related to our contribution.

[1] http://context.media.mit.edu/press/index.php/projects/

Georgia institute of technology

At the Georgia Institute of Technology (GIT) various researchers consider issues related to context computing and context-awareness. Among others, Gregory Abowd, Anind K. Dey, Bill Schilit and Marvin Theimer have conducted their studies at the Georgia Institute of Technology. The GIT contributed in the development of one of the first definitions of context-aware applications [15]. The Context Toolkit [21], developed at the GIT constitutes a major reference for context-aware architectures today. In conjunction with the Context Toolkit an application that has been developed at the GIT is the Conference Assistant [27].

An ongoing project of the GIT is the aware home project [38]. In this project, a house that is aware of its occupants' whereabouts and activities is considered and designed. The research results from this project shall serve elderly people in their everyday life.

Department of pervasive computing, Johannes Kepler University, Linz

At the department of Pervasive Computing at the Johannes Kepler University of Linz several research activities regarding pervasive computing are concentrated.

These include the concept and design of mobile and pervasive system architectures, context sensitive applications and smart systems. Furthermore, context prediction has been studied. A more detailed discussion on the latter activities is given in section 3.1.

Proposals and concepts are typically not only studied theoretically but design concepts and prototype implementations are presented that demonstrate the concepts considered. The following three examples might provide the reader with an idea of the broad scope of research conducted at the department of pervasive computing.

In [39], Alois Ferscha, Manfred Hechinger, Rene Mayrhofer and Roy Oberhauser present a decentralised approach for ad-hoc peer to peer applications. In this approach, applications are communicating over communication ports that abstract from the underlying communication protocol. The implementation enables an autonomic detection and utilisation of services and devices in the proximity of a user device. The realisation of such automatic and unobtrusive service discovery constitutes one key concept inherent to many ubiquitous computing scenarios.

Another aspect of research on ubiquitous environments is, for example, the digital aura [40]. It is a thought model for a pervasive computing device that considers presence, physical proximity, location and communication. The authors present the idea of a pervasive computing application that is capable of sensing the presence of objects or persons in the proximity of the executing device by utilising short-range communication technology. Depending on the configuration of the system, the application might inform the user when another person with similar interests is in its proximity and provide contact information to her.

Information available in the proximity of a user, that is regarded relevant, is downloaded to the device so that the user might utilise the information when it best suits her. A privacy protection scheme is proposed for the digital aura that enables the user to specify which information sources she is willing to allow the application to communicate with. For communication processes, incoming and outgoing data is distinguished between.

Furthermore, in [41], Alois Ferscha, Simon Vogl and Wolfgang Beer propose a novel representation of context that is based on the three aspects person, place and thing. Additionally, various concepts of context sensing are discussed. In particular, the sampling of states of a system on the one hand, and the sampling of changes of system states on the other hand, are considered. A framework for the sensing of gestures by orientation sensors is proposed in [42]. Along with this framework, gestures of three domain categories are discussed and can be distinguished between by the framework.

Telecooperation office (TecO), Karlsruhe

At the TecO, issues of ubiquitous and mobile computing are considered. Among others, Michael Beigl, Albrecht Schmidt and Hans-Werner Gellersen conduct their research at the TecO. The research institute has participated in various research projects that are related to context-awareness. The most well known projects are the Smart-Its (cf. section 2.1.6), the MediaCup (cf. section 2.1.5), Wearables and TEA (cf. section 2.1.6).

Technology for enabling awareness - TEA project

The Technology for Enabling Awareness [2] project was a joint project between four research institutes. The Starlab Research in Brussels, Belgium, Omega Generation in Bologna, Italy, the Telecooperation Office of the University of Karlsruhe, Germany (TecO) and Nokia research in Oulu, Finland.

The objective of the project was to develop a device that enables context-awareness for tiny mobile computers and to exploit market opportunities for context-awareness [43, 20]. The developed 'TEA Sensor-Board' acquires raw sensor data. In contrast to common approaches at that time, the choice of sensors did not cover location tracking sensors directly.

The sensors attached to the device are a light sensor, a microphone, an infrared sensor, accelerometers, a pressure sensor and a temperature sensor. Further experiments have been conducted with bio sensors. The design is flexible so that additional sensors might be utilised.

A second version of the TEA Sensor Board has been developed that is small enough to fit into an enlarged battery case of a mobile phone. The TEA2 board is equipped with 8 sensors with partially redundant features to enable the detection of measurement errors. The sensors on the board are photodiodes, microphones, accelerometers, a temperature sensor and a touch sensor. Apart from these sensors, additional sensors can be attached to the device.

Smart-Its project

The Smart-Its project[3] was led by the Telecooperation Office of the University of Karlsruhe, Germany (TecO). The project was developed in cooperation with the Lancaster University, UK, the ETH in Zurich, Switzerland and the Interactive Institute, Sweden.

The aim of this project was to integrate computation into small-scale embedded devices [44]. In the Smart-Its project, these everyday objects are augmented with sensing and communication capabilities. 'Smart-Its' are regarded as an enabling technology for building ubiquitous computing scenarios. The ability to infuse everyday objects with computing and communication capabilities is one major building block of Marc Weiser's

[2] http://www.teco.edu/tea/
[3] http://smart-its.teco.edu/

vision of ubiquitous computing. A generic software and hardware platform to enhance everyday objects with small computers has been developed. Ad-hoc networking, sharing of contexts and sensor fusion between several devices was investigated in this project. The Smart-Its system consists of RF communication, a sensor board, an application programming interface either for desktop computing or for the development of mobile services.

Equator project

The Equator project[4] is a six-year research project that focuses on the integration of physical and digital interaction.

The three main research challenges addressed in Equator are issues related to devices, infrastructure and interaction.

In this project a variety of new devices like the Bristol Cyberjacket [45] have been created that connect the digital to the physical reality. Sensing, display and power technologies have been considered. Among others, an accelerometer and GPS information are frequently used in applications developed in the project [46, 47]. Another interesting study with features of sensors seldom used for context-aware applications is [48], where the coverage and security characteristics of WiFi were explored. The work in Equator especially focused on the experiences and usability aspects of the user [49]. An application developed in the scope of this project is the load sensing by weight surfaces [36] that has already been mentioned above.

In this project, existing architectures have been reviewed and toolkits have been developed to reduce the cost of their construction. The publicly available framework Equip was developed in Equator [50]. Equip is a dynamically extensible framework for integrating C++ and Java based applications. It supports a variety of interfaces and devices.

Finally, concepts required to support the understanding of physical to digital interaction, and the required design and evaluation methods have been studied. As one part of this study, uncertainty and interpretation of contexts have been considered.

Various studies related to embedded systems have been undertaken in the course of the project [51, 52, 53].

[4] http://www.equator.ac.uk

Cortex project

The CORTEX project is an IST project. The project partners are the Trinity College in Dublin, The University of Ulm, Germany, the Lancaster University, UK and the University of Lisbon in Portugal.

The focus of CORTEX is to provide an open, scalable system architecture for real-time, context-aware applications that are composed of several entities [54].

The project has developed a programming model that supports the development of context-aware applications.

The different entities utilised in the CORTEX project are sentient objects, sensors and actuators. A sensor is defined as an entity that produces software events in reaction to a stimulus detected by some real-world hardware device.

An actuator, on the other hand, is an entity which consumes software events and reacts by attempting to change the state of the real world in some way via a hardware device.

Finally, a sentient object is a mobile, autonomous software component that can both consume and produce software events and lies in some control path between at least one sensor and one actuator. Sentient objects interact using an event based communication that supports loose coupling between sentient objects [25].

In the CORTEX project the definitions of context and context-awareness have been adapted to the sentient object model. Following these definitions, context is any information which may be used to describe the situation of a sentient object. Context-awareness is then consequently the use of context to provide information to a sentient object.

The project members distinguish between context representation that is processed by sentient objects and raw sensor data which is to be processed by a context representation component.

Aura project

The project Aura[5] was conducted at the Carnegie Mellon University, US. Aura was aimed to study issues related to distraction-free Ubiquitous Computing.

[5] http://www.cs.cmu.edu/aura

The goal of project Aura is to provide each user with an invisible halo of computing and information services that persists regardless of location [9]. Following the argumentation of the project, meeting this goal requires effort at every level: from the hardware and network layers, through the operating system and middleware, to the user interface and applications.

The focus of project Aura is situated in the design, implementation, deployment, and evaluation of a large-scale system capable of demonstrating the concept of a "personal information aura" that includes wearable, handheld, desktop and infrastructure computers.

Publications from the Aura project cover the Coda file system [55], prediction of CPU load [56] and the description of the Contextual Information Service [57].

Endeavour project

The Endeavour project at the University of California, Berkeley[6] was an umbrella project that aimed at increasing convenience in interaction with devices and other people.

A preparatory assumption of all projects contained is that a vast diversity of computing devices is commonly available. The project members abstract in their studies from storage and memory limitations and assume that arbitrary computing devices are compatible with each other over a standardised connection protocol.

The projects contained in Endeavour are, among others, the ICEBERG project [58, 59], Ninja and OceanStore [60, 61].

Within the ICEBERG project, the Clearing House architecture was developed that monitors traffic fluctuation and network performance to adapt resource provisioning. Topology and server-load information are explored for optimisation in location and routing within the OceanStore project. OceanStore strives to provide global-scale persistent data. In the OceanStore project Tapestry, a structured overlay network was developed. The Ninja project utilises network performance information for optimal automatic path selection. The project aims to develop a software infrastructure supporting internet-based applications. Several distributed data structures that implement a network storage have been proposed. Software packages developed in the Ninja project are available for download.

[6] http://endeavour.cs.berkeley.edu

2.2 Concepts and definitions

As mentioned in section 2.1, the concepts and ideas related to context-awareness that have not yet been commonly adopted among researchers even include the notion of context and context awareness itself. Since context-awareness is a comparably young research field, we find concepts and notions for which a variety of only partially redundant definitions have been given. On the other hand, several supplementing concepts are only vaguely described as, for example, the notion of high-level contexts, low-level contexts and raw data. In order to provide a stringent view on our research topics, we have to agree on non-ambiguous definitions for the concepts we utilise.

In this section we discuss those notions we adopt from recent work and further find comprehensive definitions for insufficiently defined concepts where necessary.

In our discussion we take a computation-centric view. Unlike other definitions that follow an application or service centric approach, we see the computation and processing of contexts as the centre of importance when context-aware architectures are considered. Context-aware applications ground their operation on an effective and reliable context provisioning layer. Consequently, the application benefits from improvements in this context provisioning layer. In a computation-centric approach we are more interested in methods and concepts to generate contexts than in the exact contexts that have to be generated. In a computation-centric approach the avoidance of errors is more important than the coping with and correction of errors. In a computation-centric approach we abstract from specific applications or environments and consider general process related issues.

2.2.1 Ubiquitous computing

In our view of ubiquitous computing we agree on the vision introduced by Mark Weiser in [13]. As a prerequisite to our study, we assume a world in which computation has both infiltrated everyday life and vanished from people's perception. We believe that both developments are not only possible but predefined, since computing devices continuously decrease in size and power consumption while increasing in computing power at the same time. In the vision of ubiquitous computing, everyday objects are equipped with computing power and communication interfaces in order to compute

and spread information. In our study we assume that computing is done in a ubiquitous environment, where multiple applications on stationary and mobile devices interact with one another. For ease of presentation we occasionally abbreviate the term Ubiquitous computing with UbiComp.

Several authors have observed challenges of ubiquitous computing environments. The authors of [23] for example, state increased autonomy, a dynamic computing environment, dynamic user requirements, scalability issues and resource limitations as most serious issues in UbiComp environments. Depending on the application type, further issues may be named.

We study challenges of UbiComp environments that are eminent for context prediction scenarios in section 3.2.3.

2.2.2 Sensors, context sources and features

In context-aware computing domains, the input data for applications is captured by sensors. Since several authors have varying definitions of sensors, we briefly recapitulate our notion of sensors. Basically, a sensor is a piece of hardware or software that provides information on the environment. Humans or animals are not considered sensors but might trigger and influence sensor outputs. We distinguish between hardware sensors and software sensors. Hardware sensors are physical entities that react to stimuli from the physical environment and provide a software interface to publish notification describing these stimuli. Hardware sensors might, for example, measure the temperature, the light intensity or the humidity. Further hardware sensors are, for instance, a fingerprint reader or also a computer keyboard or a mouse that monitors user input.

Software sensors are applications that react to software generated stimuli and that output a software generated notification describing these stimuli. Example software sensors are a calendar, an address book or an application a user is interacting with.

A sensor might provide various distinct aspects of a given context. Consider, for example, an audio sensor that provides the loudness as well as the number of zero crossings. These distinct aspects of context are often referred to as context features [20, 30]. Since we take a computation-centric approach, we are especially interested in the entity that provides information about a context feature.

We refer to this entity as a context source and consider context sources as atomic information sources for context-aware architectures. Context

sources are not synonymous to sensors that produce context data. One sensor might incorporate several context sources. A context source basically produces output values that are related to one specific feature of a sensor.

2.2.3 Context and context types

As we have discussed in section 2.1.1 various definitions of context have been given in the literature that are only partly redundant. We adopt the definition given by Anind K. Dey in [21] since it is most general and can be applied to all application areas relevant to our research. However, Dey explicitly intertwines context with the interaction of applications and humans or, as he states it, with users. We have a slightly wider understanding of context that is not restricted to the user-application interaction but that covers contexts of arbitrary entities.

Definition 2.2.3 : Context

> *Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object.*

Other definitions of context are too restricted to special cases to be applied in our general, computation-centric, consideration. Considering the revised definition given by Karen Henricksen, after which context is the set of circumstances relevant for the completion of a task [23], we disagree.

This revised definition differs from our understanding of context. First of all, we do not agree with the restriction of context to the set of circumstances that are of potential relevance for the completion of a task. The context driving, for example, could be partly sensed through the presence of the bluetooth ID of the car radio. However, the car radio is of no relevance considering the completion of the context driving.

In addition to the general understanding of the concept of context, a more concrete frame is required in order to be able to actually apply computations on context. We introduce the notion of a context element that utilises the definition of Dey and enhances the description to suit our needs in the processing of contexts.

Definition 2.2.4 : Context element

Let $i \in \mathbb{N}$ and t_i describe any interval in time. A context element c_i is a non-empty set of values that describe a context at one interval t_i in time.

An example for a context element that is constituted from the temperature, the light intensity and an IP address is then $c = \{24°C, 20000lx, 141.51.114.33\}$. Observe that this definition refers to an interval in time rather than to a point in time. This accounts for the fact that the information describing a context is obtained by measurements of the real world that typically require a time-span rather than a time instant in which the measurement is performed. However, the shorter the time span the more accurate a context element describes a context at one point in time. Since the values are obtained by measurements, we may assume that the count of context elements is finite.

In [62] it was suggested that the context types location, identity, activity and time are more important than other types in order to describe a context. Undoubtedly, studies that utilise these context types for context-aware applications dominate studies on other context types. One reason for this is that implications obtained from these mentioned context types seem to be intuitive to most people. However, we argue that the type of context useful for an application is inherently dependent on the application type and that this context might be ignorant of the location, identity, activity or time.

Consider, for example, an arbitrary person sitting in her room and reading a book. While this scenario appears to be tranquil when only the four context types location, identity, activity and time are taken into account, the general assessment might change with the utilisation of further context sources. If, for example, the room temperature instantly rises or the amount of methane in the air increases, the same situation then appears in a different light. Danger might be at hand and a swift reaction is required.

We therefore assume that the application defines the relevance of distinct context types. The relevance could be modified by any kind of weighting or duplicating of contexts. Since we propose an architecture that utilises contexts for arbitrary applications, we do not prefer any context type above any other. For the remainder of this thesis we do not bother about the correct and application specific weighting, but assume that the contexts utilised has been filtered and weighted according to the application needs

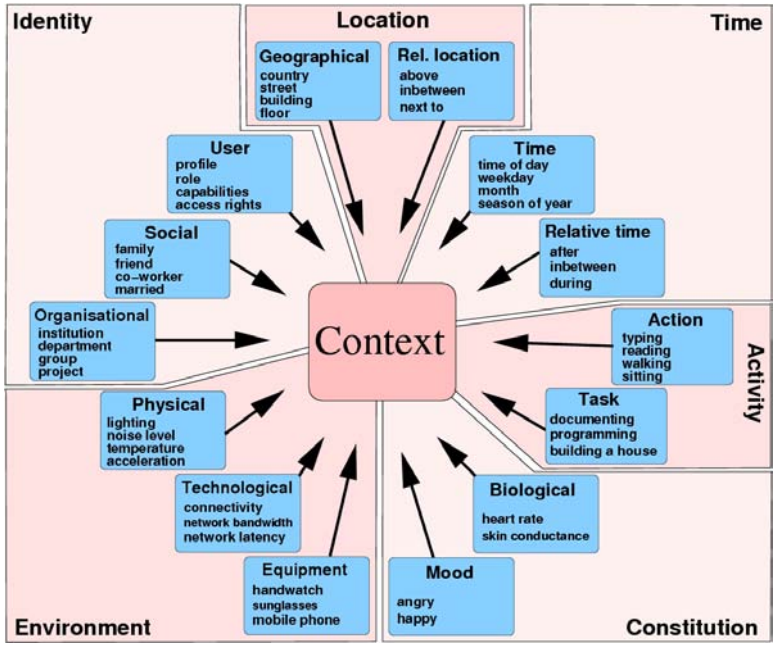


Figure 2.2: Aspects of context

in advance. Several aspects of context have been introduced in [6, 63]. A further structured and extended distinction of context types is depicted in figure 2.2. This figure should be understood as a working model of context aspects. Context specifications for the context classes depicted in the figure are examples for the context classes and can be carried on by other examples that logically fit into the corresponding context class. Further aspects of context not depicted in the figure might well be found.

2.2.4 Context abstraction levels

Context does not necessarily equal context. Two contexts of the same type that describe the same time interval might nonetheless differ from each other in value. Context has several levels of abstraction depending on the amount of pre-processing applied. A temperature context might, for example, hold the value $24°C$ as well as the value 'warm '. These context values might originate from identical measurements of context sources. However, the data abstraction level differs. The value 'warm' is at a higher abstraction level than the value $24°C$.

Although several authors use the notions high-level context, low-level context and raw data, in order to describe various context abstraction levels, no exact definition of these notions is given in the literature. These notions are therefore often used with different meanings. Some authors, for example, use the term low-level context in the same sense as other authors use the term raw data. Typically, higher context representations tend to be symbolic while lower representations are more often numeric. Generally, the definition of several data abstraction levels is reasonable since the kind of representation used for operations on context elements may affect the accuracy of the operation [64].

A rough distinction between low-level and higher level contexts is made by Anind K. Dey, Bill Schilit and Marvin Theimer [21, 15]. Following this discussion, low-level context is used synonymously for data directly output from sensors, while high-level contexts are further processed. This processing can, for example, be an aggregation, an interpretation, a data calibration, noise removal or reforming of data distributions.

Jani Mäntyjärvi further distinguishes between processed contexts that describe an action or a condition [22]. Following his notion, raw data can be, for example, $24°C$ or 70% humidity. While for low-level contexts these are further processed to conditions like 'warm' or 'high humidity'. Finally, a high-level context is an activity as, for instance, 'having lunch'.

Actually, these distinctions between high-level and low-level contexts are only required (and properly understood) by humans. From a computational viewpoint, actions and conditions are both string values obtained by further processing of raw data. From a computation-centric standpoint, both constructs are consequently on the same level of data abstraction.

High-level context	Low-level context	Raw data	Context source
walking	14°C	001001111	thermometer
walking	57.2°F	001001111	thermometer
watching movie	64dB	109	microphone
listening music	64dB	109	microphone
at the beach	47° 25.5634'N; 007° 39.3538'E	GPRMC[8]	GPS sensor
swimming	47° 25.5634'N; 007° 39.3538'E	GPGGA[9]	GPS sensor
writing	z	0x79	keyboard [en]
writing	ы	0x79	keyboard [ru]
writing	z	0x7a	keyboard [de]
office occupied	z	0x7a	keyboard [de]

Table 2.1: High-level contexts, low-level contexts and raw context data for exemplary context sources.

[8] GPRMC Example:
$GPRMC,191410,A,4725.5634,N,00739.3538,E,0.0,0.0,181102,0.4,E,A*19

[9] GPGGA Example:
$GPGGA,191410,4725.5634,N,00739.3538,E,1,04,4.4,351.5,M,48.0,M,,*45

A computation-centric approach

We therefore take an alternative, computation-centric, approach and classify the level of abstraction of contexts by the amount of pre-processing applied to the data. Throughout our work we distinguish between high-level context information, low-level context information and raw context data[7] (cf. table 2.1).

In table 2.1, exemplary raw context data, low-level contexts and high-level contexts are depicted. Note that in all data abstraction levels different context representations are possible even if the measurement is identical. An example well-suited to illustrate this is the keyboard sensor. The same key pressed on an English and a Russian keyboard (raw context data identical) might result in different low-level contexts due to an alternative language setting (acquisition procedure). In the Cyrillic layout the letter 'ы' is obtained while it is the letter 'z' for the English layout.

[7] For ease of presentation, we utilise the notions 'raw data' and 'raw context data' synonymously.

However, for German keyboards the letters 'y' and 'z' are exchanged compared to the English layout, hence leading to the same low-level context even though the raw context data is different. Furthermore, different context interpretation procedures may lead to distinct high-level contexts (office occupied or writing).

A discussion of the three data abstraction levels 'raw context data', 'low-level context' and 'high-level context' is given in the following.

The output of any context source is considered as raw data since it most probably needs further interpretation. Already at the very first abstraction level of raw context data, basic operations on the measured samples might be suggestive. Computations that might be applied on this data include mechanisms to correct possible measurement or sensor errors, filters that might abstract from irrelevant measurements or also processes that weight the measurements. Since for the remainder of this thesis we focus on context processing operations that are applied after these early data manipulation steps, we exclude all data manipulation processes applied at this pre-context stage from the scope of our research in order to avoid non-intended side effects. For the remainder of the thesis we assume that raw context data represents information measured from context sources that has already undergone these elementary data manipulation operations.

Different manufacturers produce sensors with varying output even though the sensors might belong to the same class. This is because of possibly different encodings of the sensed information or due to a different representation or accuracy. Two temperature sensors may for instance differ in the unit (Celsius or Fahrenheit), in the measurement accuracy or in the measurement range. A pre-processing of raw context data is necessary so that further processing is not influenced by special properties of the context source itself. We refer to this pre-processing as the context acquisition step. Low-level contexts are acquired from raw context data in this pre-processing step.

The data has become low-level context elements after the context acquisition. The low-level contexts of two arbitrary context sources of the same class measured at the same time in the same place is identical with the exception of a possibly differing measurement accuracy, provided that both context sources are in good order. The output of all context sources for temperature may, for example, be represented in degree Celsius.

In order to obtain high-level context elements, further processing operations are applied. Possible operations are aggregation, interpretation,

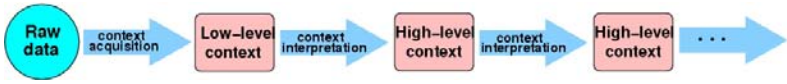


Figure 2.3: Context pre-processing steps.

semantic reasoning, data calibration, noise removal or reforming of data distributions. We refer to this pre-processing as the context interpretation step.

From low-level contexts describing the temperature, light intensity and the humidity it might be possible to infer the high-level context outdoors/indoors. There is no limit to the level of context interpretation. Several high-level contexts may be aggregated to again receive high-level context elements. For our discussion, however, we do not distinguish between high-level contexts of various context abstraction levels. For the remainder of this thesis it suffices to distinguish between the three context abstraction levels 'raw context data', 'low-level context' and 'high-level context'. For these three context abstraction levels, the distinguishing factor is the amount of pre-processing applied. Note, however, that we do not exactly define the amount of pre-processing for all three context abstraction levels since it may vary between distinct application scenarios. For our discussion it suffices that this construct of context abstraction levels is hierarchical. The amount of pre-processing applied to high-level contexts always exceeds the amount of pre-processing applied to low-level contexts in the same application scenario.

Observe that it is possible that two contexts of the same context type are differing in their context abstraction level when the amount of pre-processing to derive these contexts differs. While this might intuitively appear inconsistent, it is inherently logical from a computation-centric viewpoint. The amount of computation or pre-processing applied to contexts of distinct context abstraction levels differs. In addition, the information certitude of contexts in distinct abstraction levels might differ. We discuss this impact of context processing operations on the information certitude in chapter 5. Various context processing steps and corresponding input and output data are depicted in figure 2.3.

Vertical and horizontal processing

We distinguish between horizontal and vertical context processing operations. So far, we have considered vertical processing operation only. Vertical context processing operations alter the context abstraction level. Processing operations that do not alter the context abstraction level are considered horizontal processing operations. We consider the extrapolation, interpolation or prediction of contexts as horizontal processing operations. These processing operations are horizontal in the sense that they acquire contexts at possibly new sampling intervals rather than processing contexts in the boundaries of an existing sampling interval.

General assumptions

We assume in our work that a common application or service expects high-level context elements as input data. Except for trivial applications, low-level context is only useful for applications after a further interpretation has been applied. However, further processing on low-level contexts might well be reasonable in order to prepare the data for further operations. For raw context data, a direct utilisation of the data for applications as well as for processing steps is infeasible since this would imply that all sensor characteristics and acquisition logic has to be known by the applications or processing steps themselves. This approach is consequently only possible in small scale, static scenarios. In our work, we therefore assume a layered approach in which the application layer is separated from the context inference layer which includes the acquisition and interpretation methods.

A serious question regarding these context abstraction levels is their impact on context processing operations. The higher the context abstraction level, the more processing operations have been applied to the context elements in advance. Generally, each operation applied holds the danger of error. Contexts of higher abstraction levels are therefore potentially more likely to be erroneous than contexts of lower abstraction levels. On the other hand, it might be feasible to reduce the errors contained in a context by special purpose error correction processes. However, these error correction mechanisms are special operations that might be applied to contexts at arbitrary context abstraction levels. It seems preferable to apply an error correction after every context processing step. For simplicity, in our work we consequently assume that every context processing step is accompanied by an error correction operation. The output of any processing step

is considered error corrected. Note however, that an output that is error corrected is not necessarily error free since no error correction mechanism can provide a perfect correction in all cases.

2.2.5 Context data types

Since context is acquired from a set of heterogeneous context sources and is computed at various levels of abstraction, context processing operations applicable to one subset of contexts might be inapplicable to another subset.

As an example, consider IP addresses as context type on the one hand and temperature as another context type. Temperature contexts contain an implicit order regarding their magnitude while for IP addresses, an order cannot be provided in the same manner.

In [6] four data types have been introduced that group contexts applicable to the same mathematical operations together. Following this discussion, we distinguish context data types between nominal, ordinal and numerical categories. We omit the fourth category interval that was proposed in [6] since the boundaries of any context type (the only use for the interval category described in [6]) are provided for ordinal and numerical contexts in our case anyway.

The only operation applicable to nominal context data is the equals operation. Contexts of nominal context data type are, for example, arbitrary binary contexts, whereas symbolic context representations like, for instance, activities (walking, talking) or tasks (cleaning) are of nominal context data type.

Ordinal context data types further allow the test for an order between these contexts. Examples for contexts of ordinal context data type are physical contexts like lighting or acceleration when represented in symbolic notation like 'dark' and 'bright' or 'fast' and 'slow'.

Contexts of numerical context data type allow arbitrary mathematical operations to be applied on them. A good example for these context data types is the time. By subtraction, the time difference between two contexts of this type can be calculated.

We further consider hierarchical contexts, that are applicable to the 'subset'-operation. Similar to ordinal context data types, for hierarchical context data types an ordering of the contexts is possible. However, the order might be any kind of hierarchy as a directed tree or graph structure.

Examples for a context type of this class are geographical contexts in a symbolic representation as 'in office building' or 'in town'.

The operators applicable to one context type limit the number of appropriate context processing methods. A context processing method usually requires a minimum set of operations on contexts. In order to be processed by a processing method, all processed contexts therefore have to share this minimum set of operations. An easy solution to equalise all contexts is to abstract from all operators not applicable to the whole set of available contexts. Clearly, this reduces the already sparse information we have about the data and artificially restricts us to a smaller number of context processing methods.

Table 2.2 depicts the context data types of the context types introduced in figure 2.2[10].

Observe that the context data type is not related to the data abstraction level of contexts. Low-level and high-level contexts alike might be of ordinal, nominal, numeric or hierarchical context data type.

From one context abstraction level to the next higher one, the context data type may swap to an arbitrary other context data type. While, for instance, in the aggregation of contexts, the resulting context might likely support less operations than the operations applicable to the set of contexts before the aggregation, it is also feasible to add further operations by a mapping of contexts to elements that support these further operations.

2.2.6 Representation and illustration of contexts

We have now introduced the concept of context and have discussed context types, context abstraction levels and context data types at a rather abstract, theoretical level. For any problem domain, a good perception of the contexts and relations in this domain is at least helpful for the next step, the approach to solve the problem at hand.

A straightforward way to illustrate low-level contexts is to map them into a multi-dimensional coordinate system. This representation has first

[10]The classification of context types to context data types represents one example classification that is considered reasonable by the authors. However, a specific scenario might introduce context type classifications that differ from the values depicted in the table. The important point here is that in a given scenario the observed context data types might not be computed by arbitrary context prediction algorithms

Context type	nominal	ordinal	hierarchical	numerical
Organisational	+		+	
Social	+		+	
User	+			
Geographical	+		+	
Relative location	+		+	
Task	+		+	
Action	+			
Time	+	+	+	+
Relative time	+	+		
Biological	+	+		
Mood	+			
Physical	+	+		+
Technological	+	+		+
Equipment	+		+	

Table 2.2: Operators applicable to various context types

been considered by Padovitz et al [65, 66, 67]. Although another distinction between low-level contexts and high-level contexts has been applied, the same principle can also be applied in our case. The general idea is to represent for every time interval a low-level context element by a vector in a multi-dimensional coordinate system. Each coordinate axis represents a normalised aspect of a low-level context element.

High-level contexts are then sets of low-level contexts that are assigned a label. As we have discussed in section 2.2.4, in the context interpretation step this grouping of low-level contexts is achieved. Figure 2.4 illustrates the context interpretation step[11].

Low-level contexts are represented on the left hand side by dots in a coordinate system. On the right hand side, these low-level contexts are

[11] The figure connotes that the high-level contexts 'sleeping', 'working', 'leasure time' and 'disco' can be distinguished by the light intensity and the loudness. This labelling of high-level contexts is only for an easier understanding of the described context interpretation step. Note that the necessary context sources to accurately distinguish the mentioned high-level contexts is currently an unsolved research problem that is not solved in our work.

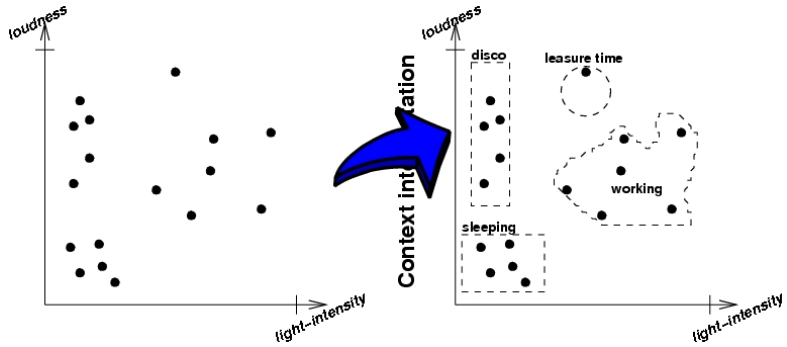


Figure 2.4: Illustration of the context interpretation step.

transformed to high-level contexts which is basically a grouping of several low-level contexts together into a set of low-level contexts.

This geometrical context representation is trivially extended to context sequences in time by simply considering one further axis in the coordinate system that represents the time. This more concrete, geometrical context representation assists us in the discussion of several properties later on.

In our discussion we do not consider overlapping high-level definitions. A discussion of this topic can be found in [67].

2.3 Summary

In this chapter we have provided a broad overview on the research field context-awareness. We have illustrated the developments in several areas related to context-aware computing. Together with general ideas and concepts we have reviewed existing frameworks for context-awareness and have given several application examples. An overview of projects and research groups related to our work has been given.

Based on these concepts, we have discussed and refined those notions that are relevant in our work. We have defined a notion of software and hardware sensors and have introduced the term context element which is grounded on Dey's notion of context. We have introduced a distinction between contexts by their data abstraction level and by the data type of the

context values. This approach differs from common proposals introduced in the literature and provides a more formal distinction between context abstraction levels. The notions of high-level context, low-level context and raw context data, that are dependent on the level of pre-processing of the data describe the distinct context abstraction levels. While we could observe a trivial relation between the number of operators available and the number of context processing operations applicable, the impact of the context abstraction level on context processing remains an open question at this point in the discussion. However, we argued that the number of operators applicable is not correlated to the context abstraction level.

As one major emphasis of this thesis we focus in the following chapters on the impact of the context abstraction level on context processing operations.

Finally, a tangible, geometrical representation of high-level and low-level contexts has been introduced that further spurs the understanding of context related issues for the remainder of this thesis.

3 Context prediction

> *The consequences of our actions are so complicated, so diverse, that predicting the future is a very difficult business indeed.*
>
> (J. K. Rowling, Harry Potter and the prisoner of Azkaban [68])

An application that is context-aware might be aware of contexts at arbitrary times [6]. Most work done on context-awareness considers present or past context. However, some authors also consider the future context (cf. section 3.1). The latter case of context computing is usually referred to as context prediction, forecasting or proactivity. While the term context prediction is most prominently used in conjunction with context-awareness [6], proactivity was originally considered for software agents. The term forecasting is most often found in relation to stochastic time series analysis. Most prominent application fields are the financial or stock market (see, for example, [69, 70]).

However, the different notions become mixed as some authors also use the terms forecasting or proactivity in order to describe the context prediction process [71, 4]. Some authors even utilise these notions in order to describe the process of inferring a context [72, 73]. To make things even more complicated, the term context prediction is not used uniformly by researchers. While the authors in [6, 4, 74] employ the term context prediction to describe an operation that infers future contexts from past and present contexts, [72] uses this term in order to describe the automatic triggering of actions when some context becomes active, while the authors of [75, 76] apply it to the process of inferring context from sensor outputs.

In our understanding, context prediction can be used by applications to

extend the knowledge about an observed context into the future. That is, to adapt their behaviour to events that will likely occur in the future. The information base on an observed context is therefore expanded by context prediction.

The cost for this additional information is an increased error probability of the predicted context. It lies in the nature of prediction that the reliability of a predicted element is typically worse compared to observed present or past events. While the impact of weak reliability may differ from application to application, this is definitely the most serious drawback to context prediction.

This chapter introduces research groups that are considering context prediction in their work. After having gained an overview of current work related to context prediction, we discuss concepts and definitions from the literature that are relevant for our studies. We further develop definitions to structure the research field where appropriate for us.

As a result of this discussion we are able to distinguish between several context prediction schemes. We also provide a first motivation for our discussion on context prediction accuracies in chapter 5. We introduce several reasons why the context abstraction level impacts the context prediction accuracy.

Furthermore, we develop a definition of the context prediction task.

3.1 Related projects and research groups

Context prediction is an evolving research field with challenging open questions. In the following sections we introduce various groups that are working on topics related to context prediction and give a brief introduction to the work they have conducted so far.

University of Exeter

In [77] Peter J. Brown and Gareth J. F. Jones study aspects relevant for context-aware retrieval of information. In their study they also discuss concepts to improve the context-aware retrieval. These are the cashing of retrieved information to reduce latency, as well as the introduction of a context diary that stores the observed contexts. While the former concept may be useful in arbitrary context-aware applications, the context diary might be used in context prediction architectures, where the prediction is

based on the observed contexts stored in the context diary. Although the authors do not explicitly mention the use of the context diary for context prediction tasks, they implicitly assume that context prediction might be utilised together with the context diary, since they assume that the diary also contains information on future contexts.

In contrast, in our approach, we do not propose to use the complete history of observed contexts, but only a subset of context sequences that is considered relevant to describe the context evolution in time.

University of Linz

A most complete overview of the research branch of context prediction is given by Rene Michael Mayrhofer (See, for example, [56, 5]). He proposes in [6] a general architecture for context prediction. In his work he defines various classes of context, analyses algorithms for clustering contexts, as well as algorithms for context prediction. Furthermore, he proposes a basic framework for context prediction which was also implemented by him. The evaluation of his, as he states it, proof of concept implementation shows that context prediction has the potential to be the first real killer application for context computing.

However, in his proposal, context prediction is based on high-level contexts only. The choice of algorithms studied does not cover recent algorithmic proposals for context prediction tasks as the state predictor [7], Self Organising Maps [78] or the alignment prediction method [8].

Mobilife project

The Mobilife project[1] was finished in 2006. Conducted by twenty two partners from industry and science, the goal of the Mobilife project was to bring advances in research on mobile applications and services within the reach of users in their everyday life. While the main focus of this project was on context-awareness, they also partly considered context prediction.

The authors of [4] also refer to Mayrhofer [6] in their approach and further discuss some issues related to context prediction. They first explain the context prediction process in more detail. According to their work, a serious problem of context prediction is that such a system may distract a

[1] http://www.ist-mobilife.org

user instead of assisting her. The authors therefore describe some applications for context prediction, where the gain of context prediction exceeds the inconvenience due to user distraction.

In their concluding discussion, they propose to utilise a hybrid server and peer to peer approach for context prediction architectures to reduce the processing load of the mobile device that hosts the context prediction architecture.

An interesting idea also introduced in this discussion is the prediction sharing paradigm. Basically the information available by various context prediction devices of members of some group is aggregated to include further knowledge into the context prediction process. They further propose to use a framework for context prediction that is as algorithm independent as possible, so that it may be easily adapted to various applications. Finally, it is proposed to represent context prediction devices as software-agents.

However, concrete approaches that solve the proposals stated are not given. In contrast to our work, the authors propose to apply the context prediction task after the context interpretation step.

University of Augsburg

At the University of Augsburg, branch prediction techniques that are common in microprocessors [79] are adapted to the task of predicting context time series. In this approach, contexts are represented as states of a state machine. For every possible context a user is in, a state machine is associated that indicates the context that most likely will be observed next. The current state of the state machine indicates the context that will be predicted. The state machine in this way always stores the context transition that was observed for the last transition made.

One can view the state predictor method as a restricted Markov prediction approach. Only transition probabilities with values 1 or 0 are allowed in all proposed modifications.

Several further publications from the University of Augsburg proposed variations and improvements to the state predictors.

The state predictor was experimentally compared to neural network predictors, to Bayesian network predictors and to a Markov prediction method [80, 81, 82, 83, 63, 84]. The authors could show that the state predictor scored respectable results compared to the adversary algorithms.

The scenarios, however, were characterised by few context changes and long idle periods in which the current context remained stable. A study of the state predictor on general scenarios is not provided.

The state predictor is applied to high-level context elements only. The authors concentrate on the design and improvement of the prediction algorithm but not on further properties of the prediction process. Furthermore it is suited for one-dimensional context time series with a low number of different contexts and a short prediction horizon only.

ISET

At the ISET institute[2], methods to estimate the short-term evolution of sampled time series is studied. The data set utilised consists of wind power samples from several wind-farms in Germany. The decision is based on the observed wind-power values for these wind farms in the context history. At the ISET institute, various prediction methods are studied and improved. Among these are artificial neural networks, a particle-swarm optimisation method and nearest neighbour approaches [85].

Since the prediction is directly based on the wind-power outputs obtained from the wind-farms, all approaches can be classified as low-level context prediction approaches.

We have also utilised the scenario of wind-power prediction in order to test various prediction approaches on realistic data (cf. section 6.4.1).

Rutgers - The state University of New Jersey

Brian D. Davison and Haym Hirsh have studied a simple algorithm to predict the next command a user types on the command line of a UNIX machine [86]. IPAM, the algorithm proposed, considers the frequency of the observed sequence, given by the last two commands observed, in the complete command sequence. The most frequent follow-up command is then predicted. Although the algorithm considered is simple, the simulation results compared to common prediction algorithms are respectable. The sensor in this approach is the keyboard utilised to type in the commands. A single letter is considered a low-level context while the aggregation of letters to commands is a high-level context. Although they do not explic-

[2] http://www.iset.uni-kassel.de

itly mention it, the authors therefore apply their prediction algorithm on high-level contexts.

In [87] Haym Hirsh and Gary M. Weiss further consider an event prediction algorithm that is applicable to non-numerical data. Their prediction method is based on a genetic algorithm that finds the event sequences that most probably indicate the expected occurrence of a specific event. The prediction method is flexible and tolerant to noisy input data. Although the event prediction problem defined in this paper is similar to the context prediction problem considered in our work, it has several different aspects. Most seriously, the output of the event prediction algorithms is a Boolean value indicating if one specific event is likely to occur at some point in the near future. For the context prediction problem, we expect a sequence of consecutive contexts that will likely occur in the near future as output. Furthermore, the event prediction algorithm is not capable of predicting the expected occurrence time of the predicted event.

University of British Columbia

The authors of [88] introduce a novel prediction algorithm which they name ONISI. ONISI utilises pattern matching approaches for context prediction. This algorithm analyses the observed sequence of contexts for a pattern that matches the sequence of most recently observed contexts.

ONISI searches for exact matches only. It is therefore hardly useful for realistic scenarios, since the observed user context patterns seldom match the typical context patterns exactly due to unpredictable random events that infer with the frequent patterns. Such an event is, for example, constituted by a colleague calling which leads to a slightly longer time in office and accordingly shifts the following context sequence for several minutes. The ONISI prediction algorithm is compared to the IPAM algorithm that was presented in [86]. Since the length of the matched pattern is shorter for IPAM, and therefore less information is contained in the observed pattern, the ONISI algorithm achieves higher prediction accuracy. The prediction of ONISI is done exclusively on symbolic contexts, ie on high-level contexts.

University of Tel-Aviv

Heikki Mannila, Hannu Toivonen and A. Inkeri Verkamo consider the problem of predicting the next outcome of an arbitrary binary sequence, which can easily be generalised to the prediction of arbitrary sequences by representing each symbol of the sequence as a binary pattern in [89].

The notion of finite state predictability is defined in the paper which is the minimum fraction of errors that can be made by any finite state predictor. The authors further provide a proof that a prediction scheme based on the Lempel-Ziv parsing algorithm can always attain finite state predictability for infinite sequences.

While this result demonstrates the power of Markov prediction schemes as the Lempel-Ziv based prediction algorithm, it is not directly applicable to context prediction scenarios. First of all, due to memory constraints, the input sequence observed by the prediction algorithm is finite. Furthermore, we do not expect arbitrary input sequences but sequences that inherit a structure that represents the individual behavioural habits. A domain specific prediction algorithm might therefore outperform the Lempel-Ziv predictor.

Prediction by Self Organising Maps

In [90], G. Simon (Université catholique de Louvain), A. Lendasse (Helsinki institute of technology), M. Cottrell (Université Paris I), J. Fort and M. Verleysen (both Université Paul Sabatier Toulouse) present an approach to context prediction with Self Organising Maps. The idea is to represent context patters in a Self Organising Map. Similar patterns are organised in a shorter distance to each other. This method is basically a pattern matching approach but its strength originates from the tolerance of small changes in the observed patterns. The extent of this tolerance threshold is guided by the parameters of the Self Organising Map algorithm.

The authors especially focus on the long-term prediction capability of their approach since several other publications already cover short term prediction with Self Organising Maps [78, 91, 92, 93, 94, 95].

The authors remark that the prediction accuracy of any prediction algorithm is increased if the context history that contains all information utilised for prediction is free of any noise. A technique to find the optimal context history is presented in [96].

The method is not well-suited for ubiquitous computing environments where the set of available context sources and contexts might frequently change or even grow. Every time a new context source is added to the system, the complete map would have to be rebuilt.

University of Colorado at Boulder

Khomkrit Kaowthumrong, John Lebsack and Richard HanThe study a prediction approach by first order Markov models in [97]. However, the prediction model presented in the paper is very limited and stationary. The adaptation to new context sources or contexts in ubiquitous environments is not considered.

The prediction is based on an analysis of the context history given by the observed high-level contexts. A generalisation to low-level contexts is not discussed in the paper.

The proposed Markov-model is only capable of predicting the type of context that will most probably occur next. Information on the actual occurrence time is not provided.

California polytechnic state University

Artemis Papakyriazis and Panagiotis Papakyriazis study a prediction algorithm for one-dimensional sequential patterns in non-stationary environments [98]. They formally state the adaptive prediction problem which describes the prediction process in a non-stationary environment. The analytical consideration in this paper is restricted to the prediction of the one-step ahead prediction probability. The authors focus especially on reducing the prediction error in this case.

University of Washington

In [99], Donald J. Patterson, Lin Liao, Dieter Fox and Henry Kautz study a method of learning a Bayesian model that describes a traveller moving through an urban environment. The location of the traveller is tracked by her GPS-coordinates. Additionally, the route the traveller will most likely take is computed by the model. Special focus of this contribution lies on inferring a user's transportation mode (ie walking, driving or taking a bus) and on predicting the transportation route.

The world model is represented by a Graph $G = (V, E)$, where the set E of edges represents the roads and footpaths and the vertices V represent intersections. The GPS positions tracked by the GPS receiver are mapped onto this graph.

The prediction procedure is therefore based on high-level contexts. A generalisation to arbitrary context prediction schemes is not mentioned by the authors.

Helsinki institute for information technology

Kari Laasonen, Mika Raento and Hannu Toivonen present a path-learning algorithm for a GSM-Network in [100]. The algorithm considers GSM-cells as atomic locations. The user-movement is tracked in a graph where the vertices are given by the various GSM-Cell-locations the user has entered so far. An edge between two vertices is drawn if the user has traversed from one of these vertices to the other. By weighting the edges of the graph, the importance or frequency of every transition is modelled. With the help of this model the authors are able to predict the probable next location of the user, given her current location and trajectory. Although the authors do not explicitly mention it, they solve a context prediction class of problem with their approach. The prediction is based on low-level contexts, since the low-level cell-IDs are not further abstracted to high-level contexts in advance of the prediction step.

A related study was also published by Heikki Mannila, Hannu Toivonen and Inkeri Verkamo [101]. The authors consider various approaches to discover frequent episodes in event sequences. This task is a preparatory step for context prediction. Once the relevant patterns in a context sequence are identified, they can be utilised by the prediction algorithm (cf. section 3.2.2).

Microsoft research

At Microsoft Research Eric Horvitz, Paul Koch, Carl M. Kadie and Andy Jacobs study a prediction system that can be utilised together with the Outlook Mobile Manager [71]. This study is a follow-up investigation of [102] where the same system short of the prediction capability was proposed. The authors focus on presence and availability forecasting. With the help of a Bayesian learning functionality, the proposed system learns

the absence times and habits of a person. This information is combined with information contained in the calendar of this person. The system then schedules incoming e-mails and voice messages according to their importance and urgency, and also might propose to colleagues that unsuccessfully call the person in question to try again at the time when she is expected to have returned to her office and is available again.

The proposed system grasps one isolated aspect of context prediction and demonstrates the usefulness of an implementation. The solution is restricted to the described application and is not applicable to other context prediction tasks. An automatic integration of new contexts or context sources in a ubiquitous environment is not discussed. The decision process for the prediction algorithm is based on classified high-level contexts. An implementation based on low-level contexts is not considered.

The MavHome project

At the University of Texas at Arlington the MavHome[3] project has been completed. The focus of the project was to develop a smart home which observes environmental information by sensors. The authors refer to the smart home as an agent. This intelligent agent can react to environmental influences and changes with the help of actuators.

An overview of the MavHome project is given in [103]. The authors describe their general design objectives and introduce the research interests of their group.

Some work done in the MavHome project also focuses on context prediction tasks. In [104] Karthik Gopalratnam and Diane J. Cook consider an incremental parsing algorithm for sequential prediction. This algorithm is based on the LZ78 family of algorithms which are originally designed for data compression tasks. The algorithm utilises Markov models for prediction of symbols in a sequence of symbols. Although this class of prediction algorithms is proved to perform well on arbitrary sequences [89], it does not utilise all information available in case of context prediction tasks. For context prediction some sub-sequences in the input sequence may contain no or even wrong information that does not describe the typical behaviour of the user. Furthermore, multi-dimensional input sequences are not considered by the authors.

[3] http://www.mavhome.uta.edu

In [105] Edwin O. Heierman III and Diane J. Cook describe a method to mine an input sequence for significant patterns. These patterns may then be utilised by prediction algorithms. The authors show that this preprocessing of input data is capable of improving the accuracy of prediction algorithms. Two prediction algorithms are considered in their study. One is the IPAM algorithm introduced in [86] while the other is a neural network approach. Both algorithms consider exact matches between the significant patterns and the observed input sequence. Small variations to the significant patterns (ie noisy input data) are not considered by the authors.

RWTH Aachen, ETH Zurich

Hans-Joachim Böckenhauer and Dirk Bongartz have published a comprehensive introduction to algorithmic basics of computational biology [106]. Although the focus of this book is not on context prediction, some concepts may also be applied to context-awareness in general.

In computational biology, DNA or RNA sequences are analysed for typical or characteristic sub-sequences. For context prediction the sequences of observed contexts can be analysed for context patterns that are characteristic for the observed context evolution. The alignment prediction approach presented in this thesis was greatly inspired by this approach.

Although the main algorithmic concept is identical, we had to consider multi-dimensional input time series and extended the algorithm by a prediction capability which were both not considered in [106].

3.2 Concepts and definitions

Context prediction introduces another variable to the context-aware scenario described in chapter 2. The concept of context prediction implicitly contains the time as one important factor of the system. With context prediction, the borderline between past and present context on the one hand, and future context on the other hand, is crossed. More exactly, past and present contexts are linked to future contexts. Observations made on past and present contexts are utilised in order to explore future contexts. Based on our discussion in section 2.2 the following sections discuss those implications to context-awareness that have an impact on context prediction.

3.2.1 Time series and context patterns

Context prediction requires the consideration of the time dimension. A set of observations $\xi_{t_1} \ldots \xi_{t_n}$ with ξ_{t_i} being recorded at a specific time interval t_i, is called a time series [107].

Note that we refer to time intervals rather than to points in time. This accounts for the fact that measurements of context sources, which are the main input source for context-aware applications, are inherently measured at time intervals rather than at time instants. For arbitrary time intervals t_i and t_j we assume that the two intervals are either identical or non-overlapping. This can be assumed without loss of generality since non-overlapping time instances can always be found for all sampled contexts.

A discrete-time time series is one in which the observations ξ_{t_i} are taken at discrete intervals in time. Continuous-time time series are obtained when observations are recorded continuously over some time interval. The authors of [17] suggest a classification of context-aware applications into continuous and discrete. In our work we are mostly concerned with discrete contexts since data is sampled at discrete points in time. If context is observed in the time domain, the concatenation of contexts measured at several times to an ordered series of consecutive contexts can be defined to be a context time series [108].

Definition 3.2.1 : Context time series

Let $i, j, k \in \mathbb{N}$ and t_i describe any interval in time. A context time series T is a non-empty, time-ordered set of context elements c_i with an attached timestamp t_i. We write T_{t_j,t_k} in order to express that the attached timestamps of the context elements in T_{t_j,t_k} are in between the beginning of the interval t_j and the end of interval t_k. We denote the empty time series with λ.

In particular, for context elements $c_{t_1} \ldots c_{t_n}$ with the interval t_i starting before the interval t_{i+1}, the time series T_{t_2,t_n} covers the context elements $c_{t_2}, \ldots, c_{t_n}$ but not the context element c_{t_1}. Observe that, since a time series contains information about the evolution of contexts in time, situation changes and even the behaviour of individuals might be described by context time series.

Context elements that share the same timestamp are grouped to time series elements.

Definition 3.2.2 : Context time series element

Let T be a context time series and t_i be a timestamp of any one context element $c_i \in T$. A context time series element $\xi_i \in T$ consists of all context elements $c_i \in T$ that share the same timestamp t_i ($\xi_i = \{c_i | c_i \in T$ and $Timestamp(c_i) = t_i\}$).
$|T|$, the length of time series T, denotes the number of time series elements in the time series T.

Note that the length of a context time series is not defined by the time difference between the first and the last time step, but by the number of time series elements. We decided on this convention for technical reasons that will become clear in later chapters. Basically, we decided for the granularity of a context time series of predefined length to be independent from the context sampling frequency.

In general, context time series therefore might contain context time series elements with more than one context element c_i. The number of context elements per context time series element determines the dimension of a time series. In a multidimensional context time series T, two context elements can share the same timestamp, wherefore the number of context elements ξ_i might exceed the number of different timestamps (ie time series elements) in T.

Definition 3.2.3 : Dimension of context time series

Let T be a context time series. T is a multidimensional time series if for $\kappa \in \mathbb{N}$ subsets $T_1 \dots T_\kappa$ exist with

1. $\forall i \in \{1 \dots \kappa\}, T_i \subset T$.
2. $|T_1| = |T_2| = \dots = |T_\kappa| = |T|$
3. *for arbitrary $i, j \in 1, \dots, \kappa$ with $i \neq j \exists c_i \in T_i$ and $c_j \in T_j$ and $Timestamp(c_i) = Timestamp(c_j)$.*

For $\Upsilon_i = \{c_j | c_j \in T$ and $Timestamp(c_j) = t_i\}$ we define the dimension $dim(T) = max_i\{|\Upsilon_i|\}$ of time series T as the maximum number of context elements of T that share the same timestamp.

An illustration of a multidimensional time series is given in figure 3.1.

Note that we have restricted ourselves to discrete events in a time series. We approximate continuous signals by taking many consecutive samples in a short period of time. The definitions above do not restrict all samples of a time series to be taken according to some fixed frequency.

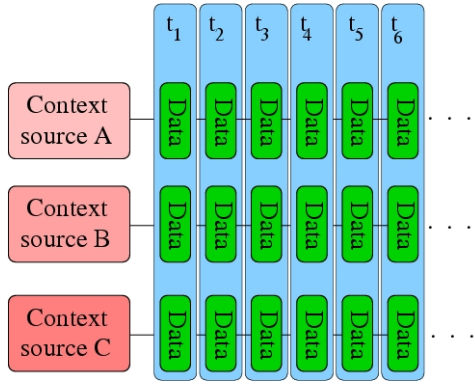


Figure 3.1: Illustration of a multidimensional time series.

We can describe the creation of a time series by a function $f : t \rightarrow T$ where t describes a point in time relative to the occurrence time of the first context element in the sequence of time series elements ξ_i and T denotes a time series. Different context patterns or time series are described by different functions $f : t \rightarrow T$. As usual, two functions $f : t \rightarrow T$ and $g : t \rightarrow T$ are differing from one another, if for any t_i the inequality $f(t_i) \neq g(t_i)$ holds.

Realistic context time series

Data sampled from one context source might differ from data obtained by another context source in sampling time and sampling frequency. Therefore, time series that are recorded in realistic scenarios can contain only part of the information on contexts of one interval in time. Since different context sources most probably generate an output value at different points in time, a time series does in realistic cases not match the simple generic pattern visualised in figure 3.1. In one time step not all context sources might produce an output value.

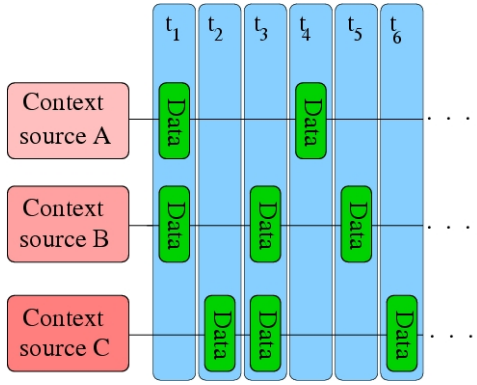


Figure 3.2: Illustration of a realistic time series.

Definition 3.2.4 : Realistic context time series

A realistic time series T is a generalisation of a multidimensional time series where any time series element $\xi \in T$ may contain one or more events of any combination of context sources.

For realistic time series, the second and third requirement in definition 3.2.3 are relaxed.

The context time series elements in a realistic time series is less symmetrical than the multidimensional time series. For every context time series element, an arbitrary number of context elements is valid.

An example of a realistic time series is illustrated in figure 3.2. With realistic time series, operations on these time series, that are otherwise straightforward become more complicated. Assume, for instance, that we wanted to compare a second realistic time series with the first one, for example in order to find similar context patterns. In most cases no sub-sequence of sufficient length can be found where the aligned entries are constructed from the same context sources considering number and type of the context sources. For an easy application of such operations to realistic time series we, interpolate all missing values in every time series or extrapolate if the missing value is younger (older) than all sampled values. However, this usually increases additional noise (errors) in the input data.

Various interpolation or extrapolation methods are suitable depending on the context data type and on application specific details. In the following discussion, we briefly discuss impacts of interpolation and extrapolation techniques for the context data types that have been defined in section 2.2.5.

Nominal contexts Since no interrelations between nominal contexts other than equality exist, a suggesting interpolation or extrapolation strategy might be to expand the context durations of adjacent contexts so that the required time instants are also covered.

Depending on the exact configuration, the context duration can be shifted to the past, to the future, or to both with varying extent. This extrapolation and interpolation approach is most useful in scenarios where contexts are considerably stable and context changes are seldom.

Alternatively, one can apply context prediction methods to obtain a context value for a given point in time. Arbitrary prediction methods might be applied in this case. This approach might also provide contexts that have been missed by the sampling process and that can therefore not be found by the interpolation and extrapolation approach. In environments where context changes are more frequent, the higher processing cost of this method might be worthwile to improve the accuracy of the observed context time series even above the accuracy of the sampled context time series.

Ordinal contexts and hierarchical contexts Additionally to the two approaches described above, the interpolation process can, in case of ordinal and hierarchical contexts, make use of the implicit order of the contexts.

Trends in the context evolution may be continued in this case. Between two contexts c and c' that are correlated by the $<$-relation, the interpolation method might insert contexts $c_1 < \cdots < c_\kappa$ which are in between considering the $<$-relation ($c < c_1 < \cdots < c_k < c'$).

An analogous argumentation does also hold for hierarchical contexts that are comparable by the $\subset$-operator.

Numerical contexts With addition of the $\cdot$ and $+$-operators the interpolation described above can be done even more advanced. Given two contexts c and c' with $c < c_1 < c_2 < c'$ we can calculate the distances $\overline{cc_1}, \overline{c_1c_2}$ and $\overline{c_2c'}$. These distances might then assist in finding the durations of these

contexts. For two output values c_1 and c_3 that were measured at time steps t_1 and t_3 with $t_1 < t_3$ we construct c_2 in time step t_2 as

$$c_2 = (c_3 - c_1)\frac{t_2 - t_1}{t_3 - t_1} \quad . \tag{3.1}$$

Low-level and high-level time series

Another distinction we regularly refer to is the difference between context time series exclusively created from high-level contexts and time series created from low-level contexts only.

Definition 3.2.5 : Low-level and high-level time series

Let T be a time series. T is called a low-level context time series if all time series elements $\xi \in T$ are low-level context elements. T is called a high-level context time series if all time series elements $\xi \in T$ are high-level context elements.

The context history

Context-aware or context prediction architectures that utilise not only present contexts, but also measurements about past contexts, need to store observed contexts for further use.

A concept that implements this is the context diary [77]. The authors propose to store contexts in a context data base called the context diary, whenever a predefined event occurs. These events are used to record all those contexts and context changes that are considered relevant. Events proposed are, for instance, a context change that exceeds a predefined threshold or user feedback that indicates the importance of the context.

We refer to the time series of observed contexts as the context history.

Definition 3.2.6 : Context history

Let $T_{0-k,0}$ be a realistic context time series of observed contexts. $T_{0-k,0}$ is a context history of length $|T_{0-k,0}|$.

3.2.2 Frequent patterns in human behaviour

Context prediction and context-awareness frequently deal with the contexts of users. In both research branches researchers implicitly assume that the behaviour of a user contains distinguishable patterns that enable

the computation of a context or even a time series of contexts. For context prediction to be feasible at least some basic conditions need to be met. Most crucial is the presence of typical patterns or at least of any reconstructable (ie predictable) process in the observed context pattern.

These assumptions have to be taken with care since the sampled contexts are only part of the definition of a certain context. Consider mood as one exemplary context of the user. The mood may have considerable influence on the way a user expects an application to react to her actions, even though it can hardly be measured by a context source [109]. Also, as the authors in [110] state, the output of context sources that lead to a specific context may change over time for one user and may even completely differ among different users.

However, reproducible, typical human behaviour patterns exist [111]. In cognitive psychology, these typical patterns are referred to as scripts. A script describes the actions and circumstances that characterise a specific context or typical context pattern. It has been shown that these scripts are similar even for groups of individuals while small alterations might exist for individuals from different cultures or societies. It could even be shown that individuals are able to complete incomplete or erroneously reported scripts so that errors in the observed sequence of contexts could be corrected [111].

These findings can be observed in various fields. As [112] states, "Behaviour consists of patterns in time". The authors of [113] for instance, observe typical behaviours in team–sport games like soccer. It is further possible to recognise the software programmer of a piece of programming code based on her programming style [114]. Some work even draws a connection between behaviour patterns and patterns found in DNA-sequences [115]. For the remainder of this thesis we assume that typical patterns exist in human behaviour and that these patterns can be described by context sequences like context time series.

3.2.3 Challenges in UbiComp environments

Context prediction in ubiquitous computing environments is seriously affected by the heavily flexible computing environment [23]. In the following sections we discuss issues in ubiquitous computing environments that do not commonly occur in other prediction domains.

Fluctuation of context sources

A key issue in ubiquitous computing environments is their changing nature. Ubiquitous computing applications access information about the environment from context sources that may be local or remote. The location of context sources that are local to an application changes only when the device that hosts the application also changes its location, while a remote context source has a trajectory that differs from that of the device. Since technologies to connect a context source with a device are only of limited range, the number of context sources available fluctuates as either the device or the context source moves.

A realistic time series of observed contexts therefore contains subsequences where context sources are available that are missing in other parts of the sequence.

This situation becomes even more complicated when a new context source enters the proximity of the device. Since the context source might provide valuable information that is not provided by other context sources, we require a context-aware application to access the new context source.

But what is the information provided by this context source? Is there any description that might provide this knowledge? Which data should be contained in such descriptions? Does it suffice to group context sources by their types (humidity, temperature) or does the application require a unique ID for every single context source or sensor? These are questions that make context prediction in UbiComp environments most challenging. Many prediction algorithms are not applicable to these environments, since they do not support such highly dynamic operation.

Adaptive operation

In ubiquitous computing we expect an environment that rapidly changes on a microscopic (single context source) level. For context prediction, a macroscopic and much slower evolution of the environment also takes place. It is the behaviour and habits of humans that gradually change with time. In some cases, external influences as a change of job or the moving to another place for other reasons might also impose sudden, drastic macroscopic environmental changes.

In order to keep a high prediction accuracy in this changing environment, an adaptive operation of the algorithm is required. A learning capability is therefore obligatory for context prediction algorithms.

3.2.4 The context prediction task

Summarising our discussion above, for context prediction, the history of observed contexts is only partially available. Furthermore, it is highly error-prone and influenced by possibly changing behaviour patterns of a user, which demands for a learning capability to be available. Additionally, the system is not closed. New contexts may enter at any time, while others may temporarily disappear. Finally, the time series analysed may contain non-numeric entries or even have mixed numeric/non-numeric contents. In search for the most suitable definition for the context prediction problem in UbiComp environments, we review definitions of related prediction problems as to their suitability in describing the context prediction problem.

Several authors have described the context prediction task from different viewpoints. However, although definitions exist for the related notions of proactivity [6, 63], time series forecasting [116, 117] and event prediction [87], we do not know of any definition of the context prediction problem.

In the following sections we briefly review definitions of other prediction problems before approaching the context prediction problem.

Proactivity

Although occasionally used in several publications as a substitute for context prediction [71, 4], the notion of proactivity in computer science is most prominently used for software agents. Proactivity in this sense is defined as taking the initiative to reach some kind of goal [118]. This definition is too wide to accurately define the context prediction task.

In [65] proactivity is defined as performing actions prior to the occurrence of a predicted situation. Since the term prediction is used to define proactivity, this notion does not lead to a more comprehensive understanding of context prediction.

The author of [6] distinguishes between proactivity and reactivity. The output of a reactive system is exclusively dependent on the past and present observations whereas the output of a proactive system may also depend on future observations. While this notion is worthwhile to provide a general description of proactive systems, a clear link to context prediction in UbiComp environments is missing. The operation on context time series clearly makes a difference to prediction algorithms since not

all prediction methods are applicable to multi-dimensional and multi-type data.

The task of learning is not covered by any of these definitions. As we have argued in section 3.2.3, learning is obligatory for context prediction in UbiComp environments. Since the user behaviour may change over time in a UbiComp environment, a static operation on the observed contexts can not be considered as prediction but as a mere reaction. Prediction implicitly strives to describe the future most accurately.

Time series forecasting

The term forecasting is usually applied in connection with numerical data in stochastic time series [116, 107, 119]. Application fields are, for example, economic and business planning, optimisation of industrial processes or production control.

A time series $T = (\xi_1, \ldots, \xi_\kappa)$ consists of κ successive observations in the problem domain. The time series T is considered to be created by some stochastic process. We assume that the process follows a probability distribution. T is considered a sample of an infinite population of samples from the output of this process. A major objective of statistical investigation is to infer properties of the population from those of the sample. In this sense, to make a forecast is to infer the probability distribution of a future observation from the population, given a sample T of past values.

The stochastic process inherent in the time series T can be described by several statistical methods. Popular models for describing these processes are moving average, autoregressive or ARMA models.

While the missing link to learning is not serious in this definition, since every single forecast is based on the recently observed time series, mixed-dimensional or mixed-type time series are not considered.

Event prediction

The event prediction problem has been defined in [87]. Following this definition, an event χ_t is an observation in the problem domain at time t. For the event prediction problem, we speak of a target event. The prediction algorithm predicts if the target event is expected at some time in the future, given a sequence of past events:

$$P : \chi_{t_1}, \ldots, \chi_{t_\kappa} \rightarrow [0, 1].$$

A prediction is considered correct if it occurs in a time window of predefined length.

This definition is especially useful in situations where single events of critical impact are predicted. An example is the prediction of a power failure of a system. Only the singular event of the power failure is of interest in this context.

Although this definition can be modified to fit the context prediction task, it then becomes quite unpractical. In event prediction the question is, 'IF' an event will occur, in context prediction we are more interested in the question 'WHICH' context will occur 'WHEN'.

Context prediction

Although some work has already been done on context prediction, no formal definition of the context prediction problem has yet been given. As stated in [6], "context prediction [...] aims at inferring future contexts from past (observed) contexts". Yet, the fluctuating nature of UbiComp environments is not covered by this definition.

The problem of context prediction can be classified as a search problem. In analogy to [120], a search problem is defined as follows.

Definition 3.2.7 : Search problem

A search problem Π *is described by*

1. *the set of valid inputs* Λ_Π
2. *for* $I \in \Lambda_\Pi$ *the set* $\Omega_\Pi(I)$ *of solutions*

An algorithm solves the search problem Π *if it calculates for* $I \in \Lambda_\Pi$ *an element* $\Omega_\Pi(I)$ *if* $\Omega_\Pi(I) \neq \emptyset$ *and rejects otherwise.*

For context prediction, the set of legal inputs Λ_Π is given by the set of legal context time series, while the set $\Omega_\Pi(I)$ is given by the set of context time series that might possibly occur in the future. The set of solutions is subject to constant changes in the observed context evolution. We call the process that is responsible for the creation of the context evolution π. The set of solutions is influenced by this process. Since the number of input parameters for the process is immense and the parameters are mostly unknown, we can assume that the process is probabilistic.

The task of a prediction algorithm is to find a context sequence in a UbiComp environment that, at a given point in time, most likely describes

the continuation of the observed context time series in the future. The context prediction task is then to find a function f that approximates the process π.

Definition 3.2.8 : Prediction quality

> *Let T denote a time series and $d : T \times T \rightarrow \mathbb{R}$ be a distance metric. We measure the quality of a prediction by the distance of the predicted context time series to the context time series that is actually observed in the predicted time interval.*

The aim of the prediction algorithm is therefore to minimise the distance to the actually observed context time series. An optimal prediction has distance zero.

Several measures of distance are feasible for various context types of the contexts contained in the observed context time series. A straightforward measure for time series represented in an Euclidean space (cf. section 2.2.6) is the sum of the Euclidean distance between the context time series elements. However, the total value of this distance measure is dependent on the length of the context time series considered.

Two metrics commonly utilised to measure the distance of two time series are the 'Root of the Mean Square Error'(RMSE) and the BIAS metric. For a predicted time series of length n, these metrics are defined as

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (p_i - d_i)^2}{n}} \tag{3.2}$$

$$BIAS = \frac{\sum_{i=1}^{n} |p_i - d_i|}{n} \quad . \tag{3.3}$$

In these formulae, p_i depicts the predicted value at time i while d_i is the value that actually occurs at time i.

Definition 3.2.9 : Context prediction

Let $k, n, i \in \mathbb{N}$ and t_i describe any interval in time. Furthermore, let T be a realistic context time series. Given a probabilistic process $\pi(t)$ that describes the context evolution at time t_i, context prediction is the task of learning and applying a prediction function $f_{t_i} : T_{t_{i-k+1},t_i} \rightarrow T_{t_{i+1},t_{i+n}}$ that approximates $\pi(t)$.

The accuracy of a context prediction algorithm can be defined with the help of the prediction quality.

Definition 3.2.10 : Prediction accuracy

For any context prediction algorithm A, the prediction accuracy is given by the approximation quality d_A if the algorithm produces predictions whose prediction quality is bounded from above by d_A.

This definition combines all parts that constitute the problem of adaptively computing future contexts in dynamic ubiquitous computing environments.

To begin with, all adverse properties of the observed history of contexts are covered by the consideration of realistic context time series. In addition, the required learning capability is included and finally, the implicit dependency on the possibly changing context evolution process is taken into account. The prediction aim is then to increase the prediction accuracy.

Note that an adaptive prediction problem has also been discussed in [98]. However, this definition is not applicable to context prediction in ubiquitous computing environments, since possibly incompletely sampled, multi-type time series are not covered.

3.2.5 Context prediction schemes

In section 2.2.4 we have introduced the distinction between raw context data, low-level contexts and high-level contexts. Regarding context prediction we have to consider at which level of abstraction the context prediction process should be applied.

Since raw context data is not yet in a consistent representation, further complications in context processing would be introduced if prediction was

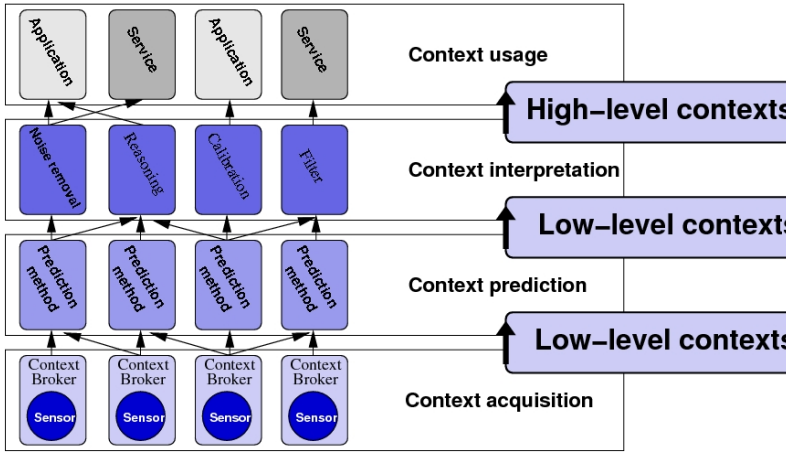


Figure 3.3: Context prediction based on low-level context elements.

based on raw context data (cf. section 2.2.4). We therefore suggest not to utilise raw context data for context prediction.

In the literature, context prediction is usually based on high-level contexts (see for instance [6, 100, 121, 86, 4, 63, 122]). The authors of these studies first interpret the low-level context to obtain high-level contexts. Afterwards, the prediction is based on the interpreted high-level contexts. This approach is appealing as long as the number of high-level contexts is low. Compared to the high number of combinations of low-level contexts from all available context sources, the set of high-level contexts in typical examples is considerably small. The requirements for the applied prediction method are therefore low. This, of course, changes if the number of high-level contexts rises in more complex scenarios.

Furthermore, the prediction based on high-level contexts has vital restrictions due to a reduced knowledge about the context itself. We therefore propose to base the prediction procedure on low-level contexts (cf.figure 3.3).

We discuss issues on the context prediction accuracy that originate from the utilisation of data at various abstraction levels in the following sections.

Reduction of information

Following our definition of context in section 2.2.3, context is any information that can be used to describe a situation. The more information available, the better a context can be described. We therefore have to be careful when it comes to context processing, since the amount of information of a context might be reduced by context processing operations.

Some of the information contained in a low-level context is lost when transformed to a high-level context since the transformation function is typically not reversible. If the obtained information on low-level contexts suffices to conclude a high-level context, we can obtain this context at any time in the future provided that the information on low-level contexts is still available. Once we abstract from low-level contexts to high-level contexts, we cannot unambiguously obtain the low-level contexts we abstracted from.

The only exception from this rule is met in the case that the interpretation step unambiguously maps one low-level context on one high-level context. Since this is then basically a relabelling process, it cannot be considered as vertical context processing step. The context abstraction level consequently remains constant. In case of a vertical context processing step the context abstraction level is altered.

Assume, for example, a setting in which context sources to capture temperature and air-pressure are given. Figure 3.4 depicts several measurements in this setting at various points in time. Every dot in the coordinate axis represents one sample obtained from both context sources. These dots are low-level contexts. The corresponding high-level contexts are 'cold/low-pressure', 'cold/high-pressure', 'warm/low-pressure' and 'warm/high-pressure'. In the figure, the air pressure and the temperature both rise. We may assume that a high-pressure area is approaching which in turn leads to better weather conditions and a higher temperature.

From the low-level samples, a general evolution of the observed process is already visible at an early stage of the observed process. However, for high-level contexts this trend remains mostly hidden.

Another example that illustrates how information contained in low-level contexts is lost by the transformation function is depicted in figure 3.5 and figure 3.6. The figures depict maps of a region in Kassel (Germany) that were generated by Google maps[4]. In the uppermost picture in figure 3.5,

[4] http://maps.google.com

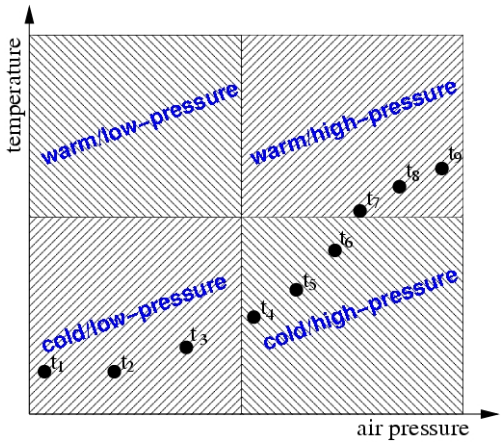


Figure 3.4: A set of measurements from context sources for temperature and air pressure.

several regions have been clustered to high-level contexts. The high-level contexts in the figure describe the locations 'Bank', 'Church', 'Market', 'Home' and 'University'.

Now, assume that a user is located inside the region that is labelled 'Home' and that we are able to track her position by her GPS coordinates (figure 3.5, picture 2). The position of the user is marked with the little dot inside the 'Home'-region. The observed time series of low-level contexts is depicted on the left hand side of the map while the time series of high-level contexts can be found to the right of the map.

If the user then starts moving inside the 'Home'-region (figure 3.5, picture 3), the low-level context of the user changes while the high-level context remains unchanged.

Only when she leaves the 'Home'-region, the high-level context also changes. However, as long as the user had entered an unlabelled region, no information about the direction of the movement can be obtained in the high-level case (figure 3.6, picture 1). Only when the user enters another classified region (figure 3.6, picture 3) will the movement direction and the new position be available.

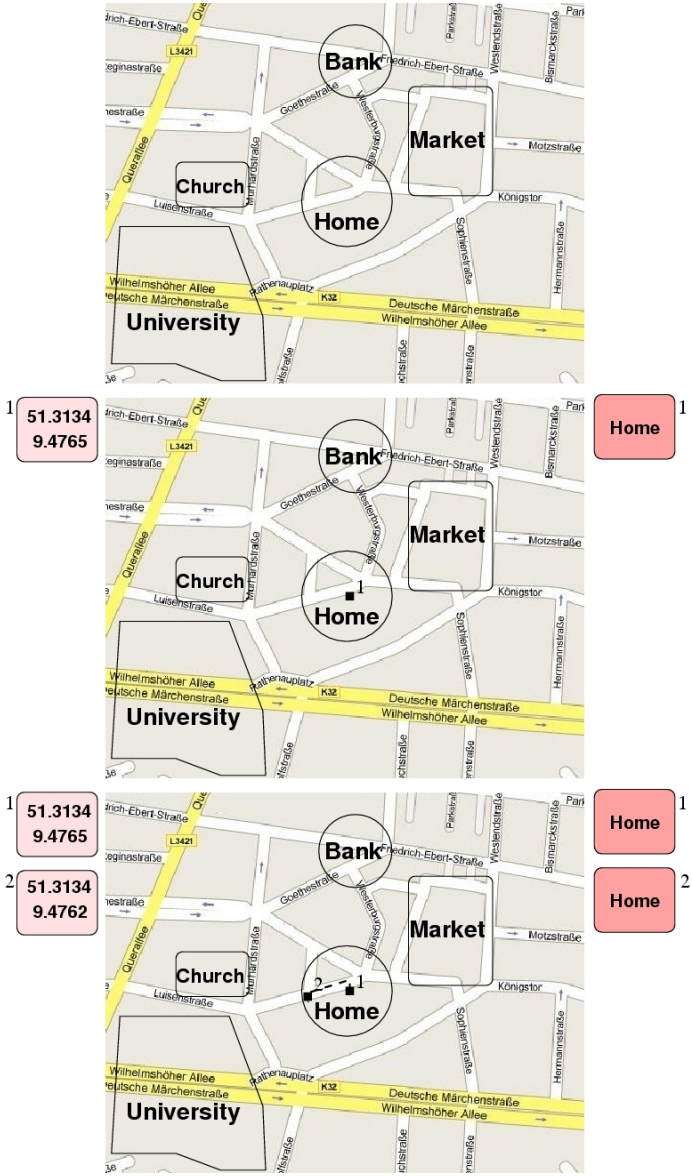


Figure 3.5: Movement of a user tracked by classified high-level contexts and by low-level contexts.

Figure 3.6: Movement of a user tracked by classified high-level contexts and by low-level contexts.

A time series of observed contexts consists of several samples from various context sources. The associated low-level contexts may indicate some general trend as it is shown in the example. However, the high-level contexts might mask this trend to some extend due to the higher level of context abstraction.

Reduction of certainty

In section 2.2.4 we have introduced several context processing steps that constitute the whole context prediction procedure. We distinguish between the context acquisition step, the context interpretation step and the context prediction step. For each of these processing steps, algorithms have to be applied, that guarantee a quality of service (QoS) regarding their memory requirements, processing loads, processing time and accuracy.

In this thesis we primarily focus on the accuracy of the overall context prediction procedure (cf. section 3.2.4). The accuracy of every single processing step is dependent on various factors as, for instance, the context data type. In analogy to the notion of accuracy for the context prediction process the accuracy of arbitrary context processing steps denotes the probability that processing errors occur in the processing step.

The accuracy of the context prediction process differs depending on the order in which the distinct processing steps are applied. This property is studied in detail in section 3.2.4.

Reduction of the long-term accuracy

Provided that a context source is in good order, the low-level contexts reflect the actual situation with respect to the measurement inaccuracy. We therefore argue that low-level contexts are of higher credibility than high-level contexts. By interpreting low-level contexts to high-level contexts, we make an educated guess based on the information that is available in the current situation. Since context is also dependent on information that is not measurable by context sources [109], we cannot exclude the possibility of misjudging the current context.

A prediction based on high-level contexts is therefore more likely applied on erroneous information than a prediction based on low-level contexts. This does not only affect the instantaneous prediction accuracy but may also affect the long-term prediction accuracy. Since the process π that is

responsible for the creation of the context evolution might change, the context prediction method should implement a constant learning procedure in order to adapt to these changes. The learning procedure is based on the observed contexts which is more likely to be erroneous in the high-level case. Hence, a constant learning procedure will be misled by this erroneous information. The long-term accuracy of high-level context prediction will consequently decrease. A more detailed investigation on this problem can be found in section 5.2.

Discussion

We have argued that the accuracy of context prediction architectures is influenced by the context prediction scheme utilised. The aspects we discussed all indicate benefits of low-level context prediction schemes above high-level context prediction schemes.

3.3 Summary

In this chapter we have reviewed projects and research groups that consider research questions related to context prediction.

We then have discussed several concepts that are required for context prediction, which has led to definitions of context time series and the context history. Along with this discussion we also identified open issues for context prediction. Among these are the consideration of realistic time series and the impact of various context data types in multi-dimensional and multi-type realistic time series. Further challenges are introduced by the UbiComp environment and are constituted by a non-continuous availability context sources and consequently the need for adaptive operation.

With these challenges in mind, we have discussed existing definitions for prediction problems and concluded that the existing definitions are not suited as a description of context prediction in UbiComp environments. Our definition of context prediction proposed in this chapter closes this gap.

Finally, we have introduced the distinction between context prediction that is based on high-level context time series elements and context prediction based on low-level context time series elements. Our discussion on several aspects of these two prediction paradigms has led to the assertion that context prediction based on low-level context elements is potentially

more accurate than context prediction based on high-level context elements. This property is further studied in chapter 5.

4 A modular context prediction architecture

> *The best prophet of the future is the past.*
>
> (Lord Byron, Letter, 28 January 1821, in [123])

The context prediction process is constituted from several intertwining operations. Consequently, these operations have to be covered by an architecture for context prediction. Since operations might be domain dependent, a general architecture should support a flexible addition and interchange of these operations. We can then view these operations as interchangeable modules that constitute the computational possibilities of the architecture.

We assume that the designer of context-aware applications chooses from a set of modules to construct the functionality she desires. We differentiate between optional and obligatory modules. Optional modules provide optional facets that are not vital for the correct operation of the architecture but might add further context sources or improve on existing ones. Obligatory modules are necessary for the correct operation of the architecture at a given task. Some modules that are necessary for context prediction to work as expected are the prediction algorithm, a learning process and the acquisition of contexts.

Several authors have considered or utilised context prediction to-date and organised the required modules in a predefined order. However, to the best of our knowledge only two architectures for context prediction have been proposed till now. Petteri Nurmi and Miguel Martin and John A. Flanagan envision the context prediction process on top of a five-layered process of context processing that consists of data acquisition, pre-processing, feature extraction, classification and prediction [4].

Rene Michael Mayrhofer embeds the context prediction process in a context processing framework that includes feature extractor, logging, classifier, labelling and prediction modules [6]. This framework is depicted as the interface between applications and context sources.

Both authors tightly knit the context prediction process with context acquisition and context interpretation processes. Context prediction in both approaches is applied on high-level contexts. However, as we have discussed in chapter 3, the decision for a high-level context prediction scheme may be suboptimal in terms of prediction accuracy.

We take an alternative approach, by providing a more flexible architecture that additionally enables context prediction on low-level context elements. The following sections introduce the basic concepts and modules of the proposed architecture. In the design of the architecture we exclude context processing tasks that are alien to the context prediction process from the context prediction architecture. By doing this we are able to provide a lean and flexible architecture for context prediction that can be integrated into highly flexible and distributed context computing environments and that is applicable to contexts of arbitrary context abstraction levels.

4.1 Requirements for context prediction architectures

Before we make an actual proposal for an architecture for context prediction on low-level context elements, we analyse the surrounding environment in order to find requirements for the prediction architecture. Most of the analysis for a ubiquitous computing environment has already been completed in section 2.2 and section 3.2. To summarise, a learning capability to adapt to a changing context evolution process is required, as well as high flexibility to cope with non-availability of context sources and possibly contradictory requirements of distinct application domains. Furthermore, we have identified the need to cope with multi-dimensional and multi-type context time series. For context prediction, some kind of context history is obligatory for various prediction algorithms. Additionally, we require the architecture to be applied for context prediction on low-level and high-level context time series alike. In the following sections we discuss these and further requirements for a context prediction architecture.

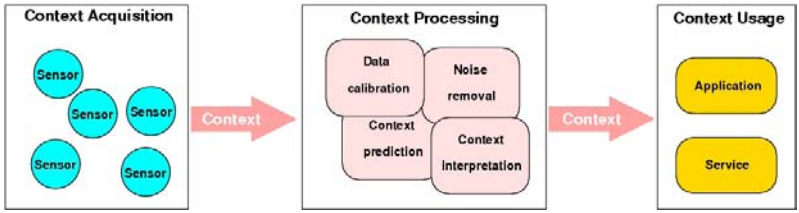


Figure 4.1: Architecture for context prediction.

4.1.1 Characteristics of context sources

Context sources attached to the architecture have characteristics that might also be valuable for context processing in the context prediction architecture. These characteristics are, for example, the position of a context source, a name, ID or class, as well as upper and lower boundaries for numerical context sources. Depending on the application and context sources at hand, further information about the context sources might also be valuable.

Since hardware sensors are typically entities of limited complexity with restricted or no memory resources, we cannot expect this information to be stored locally with every context source. A context prediction architecture therefore requires some concept that enables to store the information related to context sources.

4.1.2 Context processing

The processing of context elements is basically a hierarchical process. We divide this process into three main parts. The first is the context acquisition part in which the context elements are acquired, the second is the context processing part, that further processes the context elements and the third step is the context usage part, in which the context elements are utilised by applications and services. This structure is illustrated in figure 4.1.

The context processing part may be further divided into distinct processing steps that compute various tasks on the acquired contexts. For the scope of this thesis we roughly further shape the context processing part into two sublayers. These sublayers are the context prediction layer and

the context interpretation layer. Both layers are further described in the following sections.

4.1.3 Input and output data

A context prediction architecture expects input data in a suitable format, as well as providing output data in a predefined format. In the following sections we discuss properties of the input data expected by the architecture, along with properties of the output data provided.

A similar discussion can be found in [6]. However, the author considers context prediction that is exclusively based on high-level contexts opposed to our consideration of low-level contexts and high-level contexts for input. Hence, some requirements differ due to the different context prediction schemes utilised. We take different design decisions which consequently results in a different model.

Most importantly, we consider a different scope for our architecture. While the author of [6] describes the context prediction architecture as a black box that transforms data measured from context sources to predicted high-level contexts, we limit the scope of our architecture to the context prediction task.

Context prediction based on low-level or high-level contexts is considered as one pre-processing step (cf. section 2.2.4). Therefore, context interpretation, feature extraction and other pre-processing operations are not managed by the context prediction architecture.

Input data

Since we consider an architecture for context prediction based on low-level and high-level contexts, the input data is naturally constituted from low-level contexts or high-level contexts respectively. More exactly, a time series of low-level contexts (high-level contexts), namely the context history, is the input of the context prediction architecture (cf. section 3.2.1). Remember that the context history is potentially a multi-dimensional, discrete, multi-type context time series.

Output data

The output of the system is again a time series $T_{t_{0+1},t_{0+k}}$ of low-level or high-level context elements that constitute the prediction of contexts for

the time t_{0+1} to t_{0+k}. The prediction operation does not change the context abstraction level. Context prediction based on low-level context time series produces low-level context time series predictions and context prediction based on high-level context time series computes high-level context time series for prediction. We refer to the length of the predicted time series as the horizon of a prediction process.

Definition 4.1.1 : Context prediction horizon

Let $T_{t_{0+1},t_{0+k}}$ be the time series computed by a context prediction algorithm. The prediction horizon of this algorithm then is $|T_{t_{0+1},t_{0+k}}|$.

4.1.4 Context history

As discussed in section 3.2.1 the context history represents the time series of observed context elements that is stored by a context prediction application.

Since the availability of context sources might fluctuate, descriptions should be kept with every context pattern for a context source in the context history. Possible contents of these descriptions are the type of the context source, maximum and minimum values, a location of the context source or a classification into mobile or stationary. The actual description of the context source types is to be chosen according to the requirements of the applications that are fed by the information from these context sources.

Since the contexts of the context history represent the context time series that were observed by the context prediction application, the context history should be accessible for write or read access from the application at any time. We therefore require a secure communication link between the device that hosts the context history and the device that hosts the context prediction method. The rule base should therefore best be hosted by the same device as the prediction module.

4.1.5 Adaptive operation

As we have discussed in section 3.2.3 the highly flexible nature of UbiComp environments requires an adaptive operation of the prediction algorithm. We therefore require a learning module in the prediction architecture.

Since for different applications, different distinguishable patterns might exist (cf. section 3.2.2), the observed context time series is most likely unique. The task of the learning module is to find the context patterns that describe typical behaviour patterns. Since the observed context patterns may differ from device to device due to varying trajectories or context proximity of the devices, the learning module should be implemented on the device that hosts the application which utilises a context prediction approach.

The learning module might extract rules or functions from the context history that describe the evolution of the observed context pattern in time. There are several properties that might prove important for the description of the context evolution:

- Trends in numerical context time series
- Context patterns that repeatedly occur
- The absence or presence of specific context sources
- The length of context durations or the frequency of context changes

Table 4.1 illustrates for several context prediction algorithms, which of these properties are utilised. A discussion of the algorithms mentioned is provided in chapter 6.

For the table we observe that the nearest neighbour class algorithms as the SOM, pattern matching and alignment prediction utilise all four information sources while the other algorithms ignore at lease one information source.

In order for the architecture to store the typical context evolution patterns, we propose the use of a rule base in which rules or functions that guide the prediction process are stored. The data in the rule base represents all information learnt by the learning module. A prediction procedure might then access this information in order to provide a most accurate prediction. The format in which information about the typical behaviour patterns is stored in the rule base might differ from learning module to learning module and also between prediction algorithms.

4.2 Context prediction architecture

According to the requirements identified in section 4.1 we propose in this section an architecture for context prediction that can be applied on low-

Algorithm	Trends	Patterns	Absence/Presence	Durations
Bayesian Networks	-	+	-	-
SVMs	+	-	-	+
Markov Models	-	+	-	-
Neuronal Networks	+	+	-	-
Self organising map	+	+	+	+
Pattern Matching	+	+	+	+
Alignment prediction	+	+	+	+
ARMA	+	+	-	+
AR	+	+	-	+
MA	+	-	-	+
Kalman filters	+	-	-	+

Table 4.1: Properties of time series that are supported by various algorithms.

level context elements and high-level context elements alike.

For prediction based on low-level contexts, we insert the context prediction architecture between the context interpretation and the context acquisition layers (cf. figure 3.3). Observe however, that due to the general definition of input and output data as context time series, the architecture could also be utilised for context prediction on high-level context elements (ie between the interpretation and the application layer). In chapter 5 we utilise this property in order to compare high-level and low-level context prediction schemes in simulation scenarios.

Consequently, context acquisition and interpretation are not provided by the architecture but are utilised from alien processing parts. In the following sections we first describe those parts we utilise for preparing the input data before we discuss the modules that constitute the context prediction architecture. Section 4.3 then presents an application area for which the context prediction architecture was utilised.

4.2.1 Supplementary modules

For an architecture that is based on high-level or low-level context time series, supplementary modules are required, that provide these context

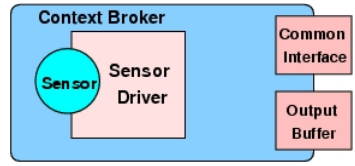


Figure 4.2: Conceptual design of a context broker.

elements. In the following sections we describe the design decisions we took for these modules in order to obtain a comprehensive structure for general context prediction tasks.

Context broker

Context sources provide measured information on contexts as raw context data. In order to acquire this raw data, a driver that serves as an interface between the architecture and the context source has to be provided.

Similar to the Context Widget approach proposed by Dey [21], we encapsulate context sources by software components that provide a uniform interface between context sources and higher layer components (cf. figure 4.2). We refer to these encapsulating components as context brokers and assume that context brokers can be arbitrarily distributed on computing devices in the UbiComp environment. The pre-eminent components a broker consists of are a driver for the context source encapsulated and a buffer containing output values for a context source. The context broker hides the specific interface for one context source from higher architectural layers and provides a standardised interface to enable dynamic discovery and removal of context sources in a flexible ubiquitous computing environment.

For communication with other context processing components, the context broker possesses a common interface. By means of this interface, a description about the encapsulated context source can be requested or configuration of the context broker is accomplished. Configuration might concern the measurement frequency, accuracies of measured low-level contexts, the size of the output buffer or specific settings for a context source. An output buffer in the context broker stores measured values for a predefined time.

Since measurements from context sources are stored in the buffer, the context broker can publish an output or a context source at any time requested by simply providing the last value measured or by extrapolation.

Characteristics of context sources are stored at the context broker. Several approaches to provide a uniform description for sensor characteristics like the SensorML language[1] already exist.

We have utilised an alternative textual description for characteristics of context sources in XML-format that is more specialised for the application domains we studied. This description can be pulled from a context broker by other modules.

We require standard tags for every context source, which uniquely identify the context source's type, boundaries, the data type and optional textual description of the context source.

Context processing

Measured context data is considered raw context data. In order to be utilised by other parts of the processing architecture, this information is transformed to low-level context elements in the context acquisition step.

The context acquisition step is situated in the context broker. In this context acquisition step, raw context data is transformed to a representation utilised by all further context processing steps. Low-level context elements are stored to the buffer of the context broker.

From there it is accessed by further processing modules. Each processing module might be constituted of several processing steps as, for instance, data calibration or noise removal operations but also context prediction.

Context processing modules that further interpret and possibly aggregate low-level contexts constitute the context interpretation part of the context processing procedure since the context abstraction level is altered by these operations. A schematic illustration of this procedure is depicted in figure 4.3.

We distinguish between horizontal and vertical context processing steps. Horizontal processing steps preserve the context abstraction level. Examples for horizontal processing steps are context prediction, interpolation or extrapolation operations.

Vertical processing steps alter the context abstraction level. Examples for vertical processing steps are noise removal, data calibration, aggrega-

[1] http://vast.uah.edu/SensorML/docs/OGC-05-086r2_SensorML.doc

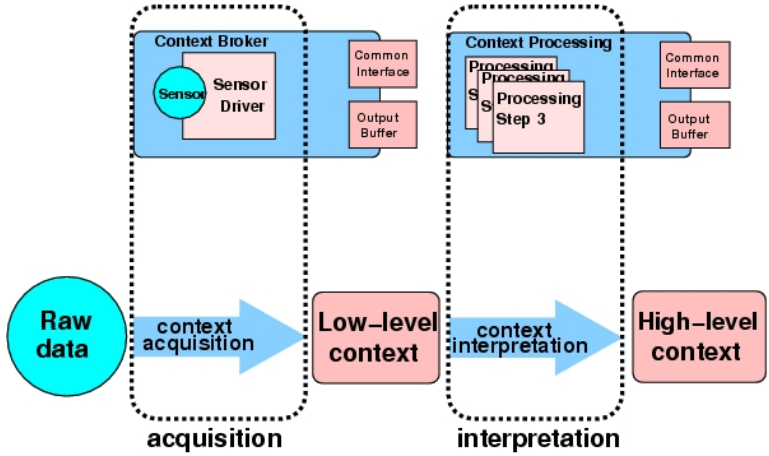


Figure 4.3: Context pre-processing steps.

tion or interpretation operations as, for example, the labelling of context elements by a context reasoning component.

All processing steps also possess a common interface for communication and configuration means and an output buffer that contains the processed context elements.

Context acquisition and context interpretation

Context information is exchanged between various processing steps in order to compute the desired result before it is forwarded to the applications that utilise the information.

An illustration of an exemplary context data-flow can be found in figure 4.4. Low-level contexts obtained from context brokers in the context acquisition layer may be further processed by context processing steps before they are utilised by applications. The horizontal context processing steps process low-level contexts or high-level contexts, while vertical processing steps interpret low-level contexts to high-level contexts or also process high-level contexts.

The figure also shows that not all data has to be utilised by every com-

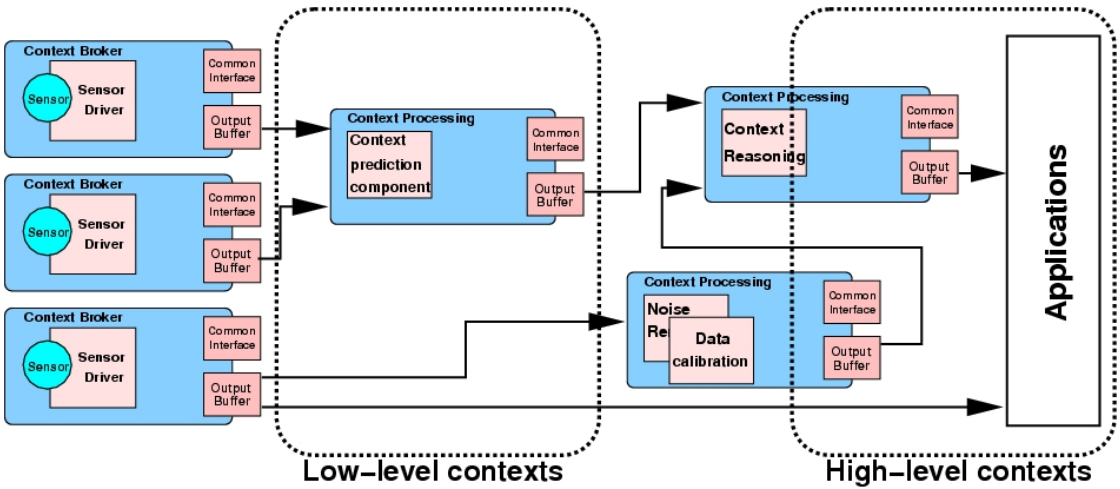


Figure 4.4: The low-level prediction component in the context data-flow in a distributed environment.

ponent. For ease of presentation we abstract from the learning component, the context history and from a rule base.

4.2.2 Flexible context prediction architecture

UbiComp environments are highly flexible and frequently changing. An architecture for context prediction on low-level context elements should therefore exploit the benefits of UbiComp environments as, for instance, the availability of a multitude of different context sources (context broker or other context processing steps) on the one hand and protect vital communication links from being interrupted due to a changing environment on the other hand. The flexibility of the environment can be supported by the use of a distributed design, as it is explained in section 2.2.2 and section 2.2.4. The processing steps communicate with each other and can be distributed in the environment. However, as discussed in section 4.1.5 and in section 4.1.4 the rule base of the learning component and the context history are inherently bound to the device that hosts the prediction application since a missed entry might seriously distort the view on the observed and typical context patterns. We therefore propose to bound these parts together in one processing step.

In this section we introduce an architecture for context prediction that is applicable to low-level and high-level context time series alike. All functional entities in the architecture are represented as processing steps which can be distributed on devices in the environment. The modules are therefore interchangeable in UbiComp environments.

The context prediction architecture is constructed from four basic modules; a learning module, a context prediction module, a rule base and a context history. We group these modules into two processing steps. The learning method, the rule base and the context history are implemented together in one processing step, while the prediction module constitutes the main part of the second processing step. These modules interact with further processing steps in the context acquisition layer and the context interpretation layer (cf. figure 4.5). The implementation we utilise in several simulations is introduced in chapter 6. The general specifications of the different modules proposed in the architecture are discussed in the following sections.

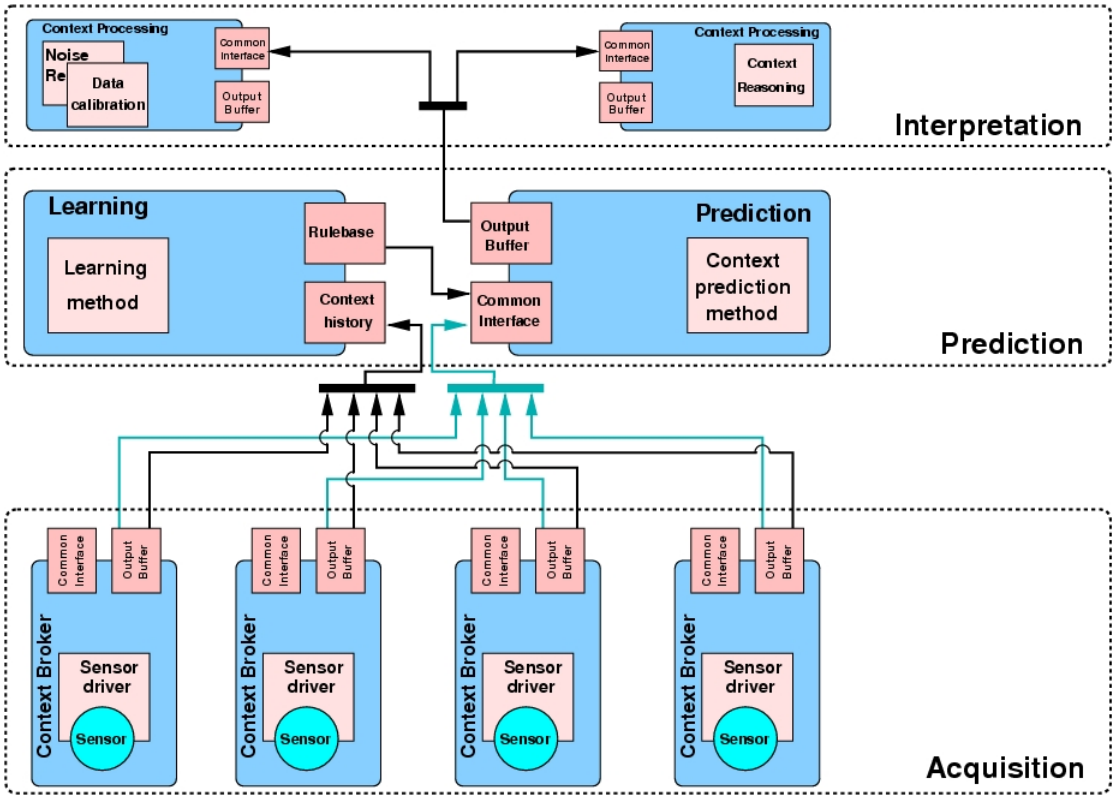


Figure 4.5: Our proposal of a general architecture for context prediction.

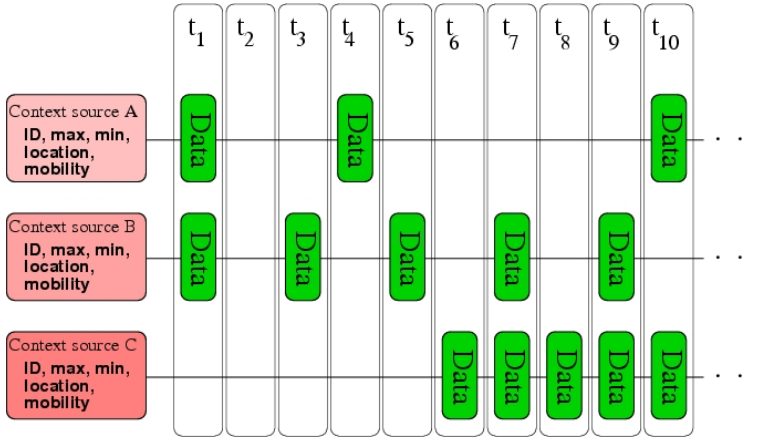


Figure 4.6: Illustration of the time series stored in the context history.

Context history

We require the context prediction architecture to implement a continuous learning procedure that adapts to a possibly changing context evolution process. We therefore require a context history that is constantly updated. The context history represents the 'memory' of the prediction algorithm which is utilised to extract rules describing the context evolution process. The context history time series is fed by all context sources available in the context acquisition layer. A schematic illustration of a possible context history can be found in figure 4.6. In the figure, the context history is composed of a set of one-dimensional time series for all context sources utilised by a context prediction architecture.

If a new context source is found, a new time series is added to this set (ie the dimension of the context history is increased by one) and if a context source is no longer reachable, it is not removed from the set but the time series is not continued until measurements from this context source are available again.

This implementation of a context history was utilised during our work. However, alternative application domains might necessitate alternative implementations.

The context history provides every aspect of context that is observable. The context sources of importance for the description of the context evolution process are computed by a learning module. A selection of context sources that contribute to the time series stored in the context history is therefore not suggestive since it bypasses the learning functionality.

For each context source attached to the architecture we track a description of this context source and a time series of low-level context elements in the context history. The maximum length of the tracked time series is dependent on preferences, implementation decisions and on memory constraints of the device that hosts the context prediction architecture.

The context history is closely coupled with the learning method since it provides the input for this method. In order to protect the communication link between the learning method and the context history as it has been discussed in section 4.1.4, we integrate the context history in one processing step together with the learning method. The context history can be considered as the interface of the learning method to the observable environmental features.

Rule base

The rule base contains the rules that the learning algorithm constructs to guide the prediction module. The syntax in which these rules are represented is dependent on the prediction algorithm that is to access the rules for prediction tasks. Depending on the prediction algorithm utilised, various possible representations of varying abstraction level are feasible for these rules.

Some examples are

- The time series of observed contexts
- Typical observed context patterns
- Transition probabilities related to context- or sequence changes
- A function describing the time series evolution

The rules in the rule base might have a weight attached in order to express the importance of the rule. As we have discussed in section 4.1, we propose to keep the rule base on the device that hosts the application which utilises context prediction capabilities since the rules generated are closely

related to the context evolution this device experiences. Consequently, we integrate the rule base together with the context history and the learning method in one processing step.

Learning component

The learning method has to support continuous learning in order to adapt to a changing context evolution process or to changing environments. The learning module therefore constantly monitors the context history and probably uses some or all of these time series to construct and update rules that are then stored in the rule base. Since we do not propose a specific implementation, the only requirements for the learning method are the interface specified by the context history and the description language for rules in the rule base which is defined by the type of the prediction algorithm.

We describe the ability of the algorithm to adapt to a changing environment with the learning frequency. The learning frequency describes how frequently the method screens the context history in order to find new prediction rules. With a higher learning frequency, the processing load of the learning module consequently increases. However, an algorithm with a high learning frequency will more often update its rule base. A rule base that is more frequently updated will in general describe the currently observed and typical context patterns more accurately. In section 5.2 we study the influence of a varying learning frequency on the context prediction accuracy.

Context prediction component

The actual context prediction is done by the prediction module. This module accesses the rule base and the momentary output data provided by the context sources in the context acquisition layer.

With the information from these two sources, the task of the context prediction method is to find the context sequence that most likely follows the most recently observed context time series. We had to take a design decision regarding the acquisition of recent contexts in the context prediction component. Either the prediction component utilises the context history of the learning component or it provides a separate history of recent contexts on its own.

We propose to provide a separate history of context evolution in the processing step of the prediction method since the prediction algorithm might require another representation for the observed context time series and since it might utilise only a subset of all available context sources, as some context sources might be considered unimportant or unreliable.

By providing a second history of contexts in the prediction component, this prediction-algorithm specific complexity is hidden from the learning component. Furthermore, with this design various prediction components might utilise the same learning component.

Prediction modules implemented In our studies we have implemented several modules of the proposed architecture. Basically, the modules can be grouped in three sets of modules that belong to three specific prediction algorithms. We have implemented a Markov prediction approach, an ARMA prediction module and an alignment prediction algorithm. In chapter 6, we discuss strengths and weaknesses of various context prediction algorithms and come to the conclusion that these context prediction approaches are best suited for context prediction tasks. The basic concepts and design decisions of these algorithms are introduced in this section.

Alignment prediction: We propose a time series based, nearest neighbour search prediction module since it is computationally not very expensive and therefore suits ubiquitous computing environments. The alignment prediction approach is novel to context prediction. The general idea has been lent from the problem of finding best alignments of string sequences. A popular application field is the search for similarities in DNA or RNA sequences. Basically, the approach tries to align two strings with the help of gaps that are inserted into the strings such that the number of matching positions in both strings is maximised. Sequences aligned are basically string sequences of a finite alphabet Σ from a computational standpoint. For each pair of letters $\sigma_i, \sigma_j \in \Sigma$ a cost function $d : \Sigma \times \Sigma \rightarrow \mathbb{R}$ provides the cost of aligning σ_i with σ_j.

While computationally expensive, it is possible to find the optimal, ie cheapest alignment for any two sequences of strings in polynomial time. With heuristic methods it is also possible to reduce the computational complexity significantly [124, 125, 126]. A comprehensive introduction to alignment methods in computational biology is given in [106].

We utilise a modification of an algorithm proposed by Needleman and Wunsch [127] to find semiglobal alignments and further expand the method to suit context prediction tasks. An analysis of the requirements and a decent description of the alignment prediction method is provided in section 6.2.5.

In this section we briefly explain the general operation of the algorithm and then focus on the architecture specific details of the prediction algorithm.

As discussed in section 3.2.2, we assume that distinct applications inherit distinct typical context patterns that can be described and observed by the context patterns contained in the context history.

The basic idea of the algorithm is to recognise the beginning of a context pattern that constitutes a typical behaviour pattern and in turn to predict the continuation of this behaviour pattern.

The alignment prediction algorithm requires a set of typical context patterns and a cost function $d : \Sigma \times \Sigma \rightarrow \mathbb{R}$ describing the similarity between two observed contexts.

The rule base for the alignment prediction algorithm.

The set of typical context patterns is provided by the rule base. The rule base therefore contains time series of context elements. It is basically a data base of time series, with additional weights describing the importance attached to each context time series. The rule base contains context patterns that are considered typical. All these context patterns are sorted by the type and number of context sources included. If the observed time series becomes identical or very similar to a context pattern in the rule base (according to the distance metric), the weight attached to the alignment in the rule base is strengthened. Otherwise the new local alignment is added to the rule base. Additionally, the weight of all context patterns in the rule base that are very different to the newly observed context pattern is decreased.

With the increasing and decreasing of weights for typical context patterns, we are able to track the gradual changes in context patterns that account for a changing context evolution process. Old behaviour patterns literary fade out in importance while new context patterns are added to the rule base. In order to control memory consumption, a simple process that removes low-weighted entries from the rule base could additionally be implemented.

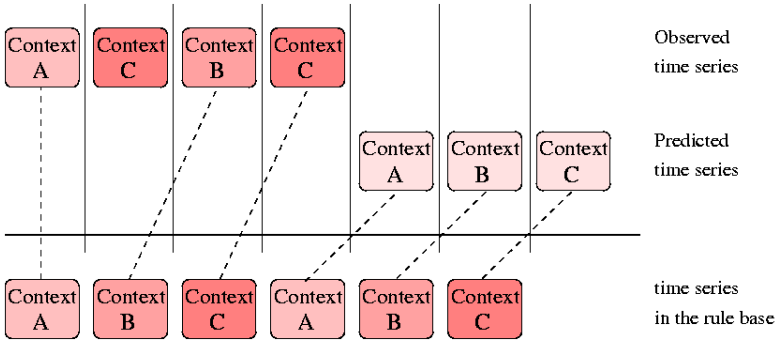


Figure 4.7: Illustration of the context time series prediction by alignment methods.

The alignment prediction procedure.

The proposed alignment prediction method is able to find similarities in the time series of observed contexts and in the rule base time series, and is resistant to a considerable number of time series entries that are missing or differing in one time series or the other. The basic prediction principle is illustrated in figure 4.7. In the figure the observed time series of the contexts A-C-B-C is aligned with every time series in the rule base. In the figure, the aligned time series is A-B-C-A-B-C. The alignment algorithm finds the alignment of minimum cost regarding the metric d. Assume, for example, the metric

$$d(\sigma_i, \sigma_j) = \begin{cases} 0, \sigma_i = \sigma_j; \\ 1, \text{else} \end{cases} . \tag{4.1}$$

In this case it is best to align the contexts A-B-C with A-C-B-C and leave a gap between A and B. The prediction of the algorithm is then the continuation of the aligned sequence in the typical context series. In the figure, the prediction is A-B-C.

According to the measure of similarity and the length of the predicted sequence, a probability is attached to each item of the predicted time series. *The learning module.*

We have implemented straight forward learning procedures that are also utilised in simulation (cf. chapter 5).

1. Keep the context history as rule base. The learning module basically synchronises the context history with the rule base. Due to memory constraints, a restriction on the time series length might be imposed. The observed time series is not further processed to remove, for example, redundant entries. The benefits of this scheme are a reduced processing overhead and the guarantee that no important patterns are missed in the rule base. However, the memory requirements are high in this case.

2. Store only important context patterns in the rule base. After an observed time series has reached a critical length of κ time series elements, the similarity of this time series to each time series in the rule base is calculated by the alignment method. If the alignment cost exceeds a similarity threshold, this time series is considered a new context pattern and is consequently added to the rule base. While the processing cost of this issue is higher than for the first scheme, the memory requirements are reduced. Furthermore, only context sequences that are considered important are added to the rule base. However, in this scheme, important context patterns might be missed. Additionally, redundant sub-sequences might be contained in the rule base time series that increase the memory requirements.

3. Store only important, not redundant context patterns. This scheme is basically identical to the second case but additionally prevent similar sub-sequences to reduce redundancy. The drawback of an increased memory requirement is solved in this case while the chance of missing important context patterns is theoretically increased: The same redundant sub-sequence might be enclosed in two different but equally important context sequences. One of these context sequences is removed when redundancy is strictly reduced.

Since we do not know which context sources are crucial to an application and since possibly several applications with different preferences access the prediction information in general, we construct local alignments out of any possible combination of context sources. Out of n patterns corresponding

to n context sources, we construct $\sum_{i=1}^{n} \begin{pmatrix} n \\ i \end{pmatrix}$ alignment patterns for any combination of different context sources. However, in the simulations presented in chapter 5, the important context sources are predefined so that only patterns containing all predicted context sources are recorded.

Credibility of rule base entries.

Each alignment in the rule base is attached a confidence value. A confidence value may range from -1 to 1. The initial confidence value is 0. If in a prediction that bases on time series T the predicted time series is not similar (regarding to a similarity threshold) to the time series that is afterwards observed, the confidence value of the time series is decreased. Otherwise it is increased. The decreasing or increasing step-width is varied with the number of correct or incorrect predictions. If the last γ predictions were not similar to the correct context time series, the stepwidth is multiplied by γ (and vice versa for similar predictions). A time series with a confidence value of -1 or below is discarded.

Markov prediction module: The idea of the Markov predictor is to utilise the frequency of context transitions for prediction purposes. We have implemented a Markov prediction approach that utilises the following implementations of the learning and the prediction method respectively.

Learning method and rule base.

The learning method utilises the context history in order to construct a Markov model in which the states represent contexts or context values and the transitions are labelled with the corresponding transition probabilities. These transition probabilities reflect the frequencies of the corresponding context changes as observed in the context history. We have implemented a dynamic prediction procedure that supports the creation of Markov models of varying order. The constructed Markov model is stored in the rule base and is updated every time the context history is updated.

Markov prediction algorithm.

The context prediction method based on the Markov approach accesses the Markov-model stored in the rule base. Starting from the sequence of recently observed contexts the model provides probabilities for possible

next contexts. In the current implementation, the most probable next context is computed by the method. In case of a prediction horizon of n the most probable contexts for the n next time steps are predicted.

Alternatively the algorithm might provide a set of most probable contexts for each predicted time step.

ARMA prediction module: The ARMA prediction approach is only applicable to context elements of numerical context data type. For the learning component, the same implementation as for the alignment prediction approach can be utilised. Regarding the learning scheme, we decided for the ARMA approach on the first and simplest method that synchronises the context history with the rule base.

We decided to take this approach since it best complies with the ARMA operating principle. ARMA algorithms basically extrapolate an observed time series with the help of autoregressive and moving average methods.

The ARMA prediction algorithm utilises the time series stored in the rule base and computes the continuation of this time series by applying autoregressive and moving average filters.

4.3 Prediction application

Context prediction architectures foster the straight-forward integration of context prediction features into applications. In the course of the project MIK21[2], the mobile event guide (MEG) that incorporates context-aware and, in particular, context prediction related features has been developed. The general idea of this client/server application is to guide visitors of large, distributed, events by reviews of former visitors.

An event can be, for example, an exhibition, a festival or also a fixed installation as, for instance, a park or zoo. We assume that an event is composed of several sites at various locations and that each of these sites offers several items. How an item is actually defined, depends on the type of an event. In case of an art exhibition, items are, for example, sculptures or drawings. In case of a festival, an item might be a musician or a group of artists, while in case of a zoo or park, distinct animals or flowers might constitute items. The idea of the mobile event guide is to

[2]MIK21: Migrationskompetenz als Schlüsselfaktor der Ökonomie des 21. Jahrhunderts, http://www.mik21.uni-kassel.de

provide a platform through which a visitor of an event can benefit from the experiences of other users. Apart from the initialisation of the system, the MEG is an unsupervised approach that evolves with the contribution of its users.

Users can rate items, take pictures of them or provide a comment on an item. These so-called recommendations are tagged with the user's current location in order to link items to sites of an event.

Given these recommendations, a visitor with the application running on his mobile device might then obtain recommendations for his stay at the event. The recommendations are filtered by the rating of other users and by the distance to the item.

Additional contexts and profile data are incorporated in order to improve the recommendation of the system. In the implementation of the mobile event guide we also utilise the weather as well as the interests and occupation of the visitor for recommendation purposes. We further consider a prediction capability for the MEG application. The prediction functionality extends the application by a route recommendation ability. The application might propose a route that starts from the current location of a user and that extends from there over several event sites that cover those items that are best rated by users with a similar profile. This proposal is based on the ratings of former visitors and on the paths they took between the event sites. The prediction component predicts the route that is expected to be most satisfying for the mobile user.

In order to distinguish the location of the user, the application utilises GPS and GSM/UMTS cell-ID. In cases where no GPS is available as, for instance, when the user is indoors, the cell-ID information is utilised for positioning purposes. All information requested, as well as recommendations given, is filtered by this location information in order to provide only information on sites and items in the proximity of the user that is expected to be of high relevance to her.

4.3.1 Application scenario

We have developed a prototype implementation of the mobile event guide for the Documenta 12[3] in Kassel, Germany. The Documenta is an exhibition for modern art that takes place in Kassel every five years and that attracts visitors and artists from all over the world. The exhibition

[3] http://www.documenta12.de/

sites are spread over the city area of Kassel. In 2007, the Documenta consists of six Documenta sites in Kassel that host numerous items each. Additionally, some pieces of art are located outdoors at scattered, isolated locations in Kassel. The various Documenta sites are considered as sites of the event. The pieces of art that are located at these sites constitute the items of the mobile event guide. We distinguish between the item categories 'sculpture', 'drawing', 'performance', 'installation' and 'video'.

4.3.2 Implementation of the mobile event guide

The MEG application follows the client-server paradigm. Most of the computation is done by a server, while clients generate the requests for the server and serve as interfaces for the users. The following sections briefly describe basic design decisions of the MEG client and server implementations.

Client

The MEG client is written in Python, in particular, the PyS60 programming language has been utilised. The mobile event guide client is executable on mobile phones with the Symbian S60 operating system. With the client software, a user can obtain information related to the event, receive recommendations which items are most interesting to her or write a review to an item she currently visits (cf. figure 4.8(a)). The MEG client serves as an interface to the server. It provides all options available and displays requests and responses from the server. GPS location information is obtained via bluetooth. We implemented a bluetooth interface to an external GPS receiver. However, the client implementation is modular, so that it can be easily extended by further interfaces. At the current state of the implementation, the user can rate an item on a scale from 0 to 9, write a description for an item and append a photo. All this information is then sent to the MEG server (cf. figure 4.8(c)).

Server

The connection between server and client is established over the http protocol. The server hosts all data from the clients in a MySQL data base. All essential computation is completed at the server side. The client only displays the options possible and the results from a server request.

(a) The main menu. (b) Information related to one event. (c) Review of an item.

Figure 4.8: Screenshots of the mobile event guide.

In the current design, the prediction architecture is also hosted at the server side. The context sources for the prediction approach are given by distinct entries of the MySQL data base. Since each entry is tagged with a location and a time stamp, time series of contexts can be obtained from the data base that are then utilised in the prediction component.

4.3.3 Evaluation

At the time of writing this thesis, the MEG application is evaluated in field tests by probants at the Documenta 12. We equipped several users with mobile devices on which we installed the program along with GPS modules. In the evaluation, we focus, apart from the acceptance and user experience, on usability aspects and on privacy related issues. Furthermore, the accuracy of the context prediction approach is evaluated. The Documenta 12, and consequently the field test, lasts from June 2007 to September 2007. Since the evaluation is not finished yet, no results are provided in this thesis but can be expected in future publications.

4.4 Summary

In this section we have discussed requirements for an architecture for context prediction on low-level contexts in UbiComp environments. We have

identified the support for a distributed design as well as an adaptability to the changing environment as main design issues. Consequently, we have proposed a modular design that integrates between the context acquisition and the context interpretation layer. Alternatively, the context prediction architecture might also be applied on top of the context interpretation layer on high-level context time series.

Our architecture is composed from two functional, loosely coupled entities that contain the necessary methods. These are a learning module and a prediction module. Both contain all functions required to state a prediction on contexts of arbitrary context abstraction level. Furthermore, we have specified the modules in more detail and proposed sample implementations for the modules. Among the implemented modules are several prediction components, namely a Markov prediction component, an ARMA prediction component and an alignment prediction component.

Although not considered in the course of this thesis, a mixed input sequence of high-level and low-level contexts can also be coped with by the architecture proposed. However, since context histories with contexts of various context abstraction levels are not in the focus of our research for this thesis. We therefore leave this feature of the architecture as a remark only.

Finally, an application scenario was presented, in which context prediction features were integrated in a general context-aware application that was implemented in the course of the thesis.

5 Results and studies on context prediction

> *But who wants to be foretold the weather? It is bad enough when it comes without our having the misery of knowing about it beforehand.*
>
> (Jerome K. Jerome, Three men in a boat, Chapter 5 [128])

Context prediction constitutes a capability that enables novel device and application behaviour in a multitude of application fields. However, the greatest hindrance to a broad utilisation of context prediction in everyday life is a property that is inherent to the nature of context prediction itself. A prediction, regardless how elaborate, is always a guess. Therefore, when considering context prediction, one has to expect erroneous predictions and cope with them accordingly.

We study issues on the accuracy of context prediction in this section. Although the reader should already feel familiar with our notion of accuracy, we would like to attract attention on an interesting property of accuracy that might not be intuitive from the start. An inaccurate prediction is not implicitly an incorrect prediction. On the contrary, an inaccurate prediction may be perfectly correct. As an example consider a prediction that was made in the year 1949:

Popular Mechanics, 1949:

"Computers in the future may weigh no more than 1.5 tons."

This can be considered as a prediction. Furthermore, the prediction is correct. However, from today's perspective we are not quite satisfied with

the prediction. This is because the accuracy of the prediction is low. First of all, the term 'computers' is imprecise. Certainly, there are very large computers or clusters of computers that operate in collaboration and that are even heavier than 1.5 tons, but when we think of computers we are first let to think of an analogue to our desktop machine at home. Considering the latter kind of computers, we can surely say that it weighs far less than even 1 ton. Moreover, the time future is too inaccurate to be really useful. There are a lot of yet unaccomplished challenges that are totally reasonable at one point in the future. Consider, for example, large-scale quantum computers. When the time of occurrence is too imprecise, predictions cease to be useful.

This section considers aspects that influence the accuracy of a context prediction algorithm. In particular, we consider differences in accuracy between high-level and low-level context prediction schemes. We first analyse the prediction scenario for low-level and high-level context prediction schemes and discuss requirements necessary for generality of this discussion.

The remaining part of section 5.1.1 focuses on discussing the results obtained for the prediction accuracy from several viewpoints. First, implications are pointed out and are further illustrated by probability graphs. Additionally, results regarding the prediction accuracy that are obtained in various simulations are presented and discussed.

Section 5.2 discusses the learning and long-term accuracy. Again, the discussion is opened by analytic considerations followed by simulation results that back the conclusions gained from the analytic argumentation.

In both sections we also conducted simulation results, since the analytic considerations do not cover all aspects of real scenarios, but necessitated abstraction from some details in order to provide a general insight into the problem domain.

5.1 High-level and low-level context prediction accuracy

In this section we consider the accuracy of two context prediction schemes. The accuracy is given by the amount of inaccurate context elements in the whole sequence of predicted context elements. We consider a predicted context element as inaccurate, if it differs from the actually observed context elements (cf. section 3.2.4).

We focus exclusively on the context prediction accuracy. Accuracy altering operations that are applied before or after the context prediction are omitted. This includes all horizontal processing operations that are applied on raw context data as well as all vertical and horizontal processing operations that are applied on predicted high-level context elements.

We distinguish between two context prediction schemes. These are the context prediction based on high-level context elements and context prediction based on low-level context elements. Both context prediction schemes are illustrated in figure 5.1.

The prediction accuracy may be decreased when prediction is based on high-level context elements. This is due to the different sequence in which the context prediction and context interpretation steps are applied (cf. section 3.2.5). In the following sections we consider the context prediction accuracy of high-level and low-level context prediction schemes. We first analytically study the prediction accuracy and afterwards provide simulations on synthetic and real data sets that support the obtained insights.

5.1.1 Analytical discussion on the impact of processing errors

Several reasons may account for a differing accuracy between the high-level and low-level context prediction schemes. Serious influences on the context prediction accuracy originate from a different error probability of the input data and a disparate context abstraction level. In the following sections we discuss the impact of the order of the applied processing steps and the impact of higher context abstraction levels on the context prediction accuracy.

For the analytical discussion of the context prediction accuracy we take several assumptions on the context elements and on the prediction process that are stated in the following.

Measurements are represented as distinct samples: A sensor measurement represents a probable assumption on the actual value of a measured quantity and can therefore be seen as a probability distribution that is centred around the measured value. An alternative approach is to represent this probability distribution with the actual measurement since it constitutes the most probable value. In our discussion we take the latter approach.

Raw context data is already processed: As stated in section 2.2.4, raw context data might have undergone various horizontal processing op-

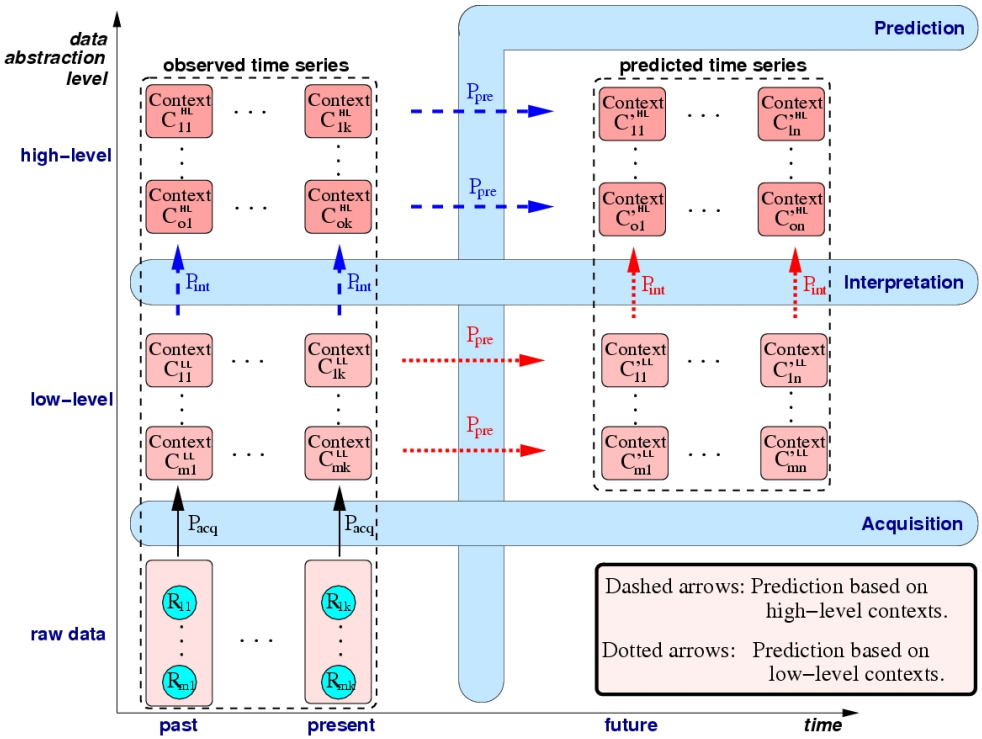


Figure 5.1: Low-level and high-level context prediction schemes.

erations so that it is in general not synonymous to distinct output values of context sources. We assume that the number is constant for all time intervals considered.

Context acquisition preserves dimensionality: Context acquisition is applied to obtain low-level context from raw context data. We assume that context acquisition is an m to m operation. For every single raw data value, a separate context acquisition step is applied that computes exactly one low-level context. Consequently, the number of past and present low-level contexts considered is identical with the number of past and present raw context data.

Context interpretation relaxes dimensionality: Context interpretation is applied to obtain high-level contexts from low-level contexts. Interpretation is applied in one time interval. Context interpretation is not applied overlapping or combining time intervals. However, context interpretation might alter the number of contexts considered, so that the high-level context time series dimension differs from the corresponding low-level context time series dimension.

Context prediction preserves dimensionality: Context prediction is applied on high-level or low-level context time series and preserves time series dimension. We model a q-dimensional time series prediction by a q-fold one-dimensional prediction.

Error probability known and constant: For every processing operation, namely acquisition, interpretation or prediction, we assume that the probability, to apply the operation without error, is known. Furthermore, we assume that the probability for each processing step is constant for each application of the operation and that the probabilities for the distinct operations are independent from each other.

Processing operations are identical: In order to preserve comparability, we assume that acquisition, interpretation and prediction operations utilised for high-level and low-level context prediction schemes are identical regardless of the prediction scheme utilised. If any of these operations is composed of several sub-operations, we assume that these sub-operations are applied in the same order for both context prediction schemes. As a consequence, the dimension of the time series elements is constant in one context abstraction level.

Number of context values is fixed: We assume that the number of possible context values is constant among context types of one context abstraction level. This is a simplifying assumption that is necessary in

order to provide a general, scenario independent discussion.

Uniform probability distribution: For errors that occur in the interpretation or prediction steps we assume an independent and identical distribution. Hence, if an error occurs, any possible error has the same probability.

Depending on the scenario in which the context prediction methods are applied, also other probability distributions are possible. However, since we aim to provide general, not environment-dependent results, we assume that all errors occur with equal probability.

Interdependency between context sources: Interdependencies between context sources are not directly modelled in the following discussion. Observe, however, that this poses no limitation to generality of the model since an existing interdependency can already be resolved in the definition of a context source. The temperature and pressure of an entity, for example, impacts its volume. A function of temperature and pressure might describe this impact. We assume that all such interdependencies are resolved by the consideration of appropriate functions in the computation of raw data of a context source.

No mixed-abstraction level prediction: We restrict the context prediction algorithm to utilise contexts of one context abstraction level only. In practise, the joint utilisation of various context abstraction levels is also feasible. Since we, however, aim to provide results on the benefits of distinct context abstraction levels, the consideration of multiple abstraction levels in one prediction is not convenient in our case.

Parameters utilised in the discussion: The high-level and low-level context prediction process differs in the order in which the context processing steps are applied. Figure 5.1 schematically illustrates this property. For high-level context prediction the context interpretation step is followed by the context prediction step, while for low-level context prediction the context prediction step is applied in advance of the context interpretation step.

For the following discussion, assume $i, k, m, o \in \mathbb{N}^{\setminus 0}$. In the context prediction process, several sources of error may be identified. These are the context acquisition step, the context interpretation step and the context prediction step. The probabilities that no error occurs in any of these context processing steps are

P_{acq} The probability that no error occurs in the context acquisition step.

P_{int} The probability that no error occurs in the context interpretation step.

P_{pre} The probability that no error occurs in the context prediction step. $P_{pre}(i)$ expresses the probability that no error occurs in the prediction of the i-th context.

We assume that the context prediction method bases its calculation on k context elements from the context history, regardless of the type of the context elements input into the algorithm (high-level or low-level). Assume that each element in the low-level context history is composed of m low-level contexts and that each element in the high-level context history is composed of o high-level contexts. In each context processing step, context elements are expected as input and are also provided as output in an aggregated or interpreted form. We define an error in one of these context processing steps as an incorrect interpretation, prediction or aggregation of context elements. An error is therefore an incorrect context element received after one of the mentioned steps has been applied. This includes erroneous context types, as well as erroneous values of one context type. In the context interpretation step, for instance, if the correct interpretation of the i-th context at time j is 'context A of value 10.0', possible incorrect interpretations are 'context B of value 10.0' but also 'context A of value 8.5'.

We assume that for $v_l, v_h \in \mathbb{N}$ a low-level context may have one of v_l different values while a high-level context may take one of v_h different values. The number of different configurations for a context time series element of the context history at one interval in time is therefore v_l^m in case of a low-level context time series and v_h^o in case of a high-level context time series.

To show the difference in accuracy, we derive the probability that an arbitrary predicted time interval is accurate for low-level and high-level context prediction schemes.

This discussion may then be generalised from single arbitrary predicted time intervals to whole predicted time series.

High-level context prediction

For high-level context prediction, the context acquisition step is the first processing step applied to the sampled contexts in form of raw data. For

all k time series elements in the context history, every one of the m raw data values is transformed to low-level contexts in the context acquisition layer of a context prediction architecture. Since P_{acq} describes the probability that no error occurs in one of these context acquisition steps, the probability that no error occurs in any of the $k \cdot m$ context acquisition steps is P_{acq}^{km} consequently.

In the context interpretation layer, the m low-level contexts of every one of the k context time series elements in the low-level context history are interpreted to o high-level contexts that constitute a time series element of the high-level context time series. Altogether, $k \cdot o$ context interpretation steps are applied in the interpretation layer. Since P_{int} describes the probability that no error occurs in one of these interpretation steps, the probability that no error occurs in the whole context interpretation process is consequently P_{int}^{ko}. Finally, $P_{pre}(i)$ describes the probability that the prediction of the i-th context is without error. Since the i-th time series element consists of o context elements (o context elements share the same timestamp), $P_{pre}^{o}(i)$ is the probability that no error occurs in the context prediction step. Together, with probability

$$P_{hl}^{approx} = P_{acq}^{km} P_{int}^{ko} P_{pre}^{o}(i) \tag{5.1}$$

no error occurs in any of the context processing steps utilised for the prediction of one specific high-level time series element. In this calculation we did not take into account that an error in the context interpretation step might correct an error that occurred in the context acquisition step, or that a context prediction error has a correcting influence on erroneous high-level contexts. The probability P_{cor}^{int} that an error which occurs in the context acquisition step is corrected by an error that occurs in the context interpretation step is

$$P_{cor}^{int} = (1 - P_{acq}^{m})(1 - P_{int}^{o}) \frac{1}{v_h^o - 1}. \tag{5.2}$$

In this formula, $1 - P_{acq}^{m}$ is the probability that an error occurs in one of the m context acquisition steps that are related to one context time series element and $1 - P_{int}^{o}$ describes the probability that an error occurs in one of the o context interpretation steps. However, in this case, no arbitrary error is required but the one interpretation error that leads to the correct high-level context. Since v_h high-level contexts are possible for every one of the o high-level contexts in one time series element, the

number of possible high-level time series elements is v_h^o. Consequently, the number of possible errors is $v_h^o - 1$ since one element represents the correct interpretation that is without error. With probability $\frac{1}{v_h^o - 1}$ the required specific error required for a correction is observed out of all $v_h^o - 1$ equally probable interpretation errors.

We now consider the probable correcting influence of the context prediction error. Since we have assumed that every one of the $v_h^o - 1$ incorrect time series elements is equally probable for any incorrectly predicted position i in a predicted time series, the probability, that the correct time series element at position i is predicted from an erroneous context history is $\frac{1}{v_h^o - 1}$. Altogether, the probability $P_{hl}(i)$ that time series element i is accurately predicted if the prediction is based on the high-level context time series is therefore

$$\begin{aligned} P_{hl}(i) &= \left(P_{acq}^m P_{int}^o + P_{cor}^{int}\right)^k P_{pre}^o(i) \\ &+ \left(1 - \left(P_{acq}^m P_{int}^o + P_{cor}^{int}\right)^k\right) \frac{1 - P_{pre}^o(i)}{v_h^o - 1}. \end{aligned} \tag{5.3}$$

Note that we consider interpretation and acquisition errors separately for every one time series element. This is expressed by the exponent k which affects the term $P_{acq}^m P_{int}^o + P_{cor}^{int}$ as a whole.

In analogy to this discussion, we receive the probability that a predicted high-level time series T of length $|T| = n$ contains no inaccurate context time series element as

$$\begin{aligned} P_{hl} &= \left(P_{acq}^m P_{int}^o + P_{cor}^{int}\right)^k P_{pre}^o \\ &+ \left(1 - \left(P_{acq}^m P_{int}^o + P_{cor}^{int}\right)^k\right) \left(\frac{1 - P_{pre}}{v_h^n - 1}\right)^o. \end{aligned} \tag{5.4}$$

In this formula, P_{pre} depicts the probability that a one-dimensional time series of length n is correctly predicted. Since the dimension of the predicted time series is o, P_{pre}^o describes the probability that this o-dimensional time series is error free. The term $\left(\frac{1-P_{pre}}{v_h^n - 1}\right)^o$ depicts the probability that an error in the interpreted time series that occurs with probability

$$\left(1 - \left(P_{acq}^m P_{int}^o + P_{cor}^{int}\right)^k\right) \tag{5.5}$$

is corrected in the prediction step. Again the prediction errors are considered separately for every dimension of the high-level context time series, since the exponent o affects the term $\frac{1-P_{pre}}{v_h^n-1}$ as a whole.

Low-level context prediction

For low-level context prediction, the context prediction step is applied in advance of the context interpretation step. Consequently, with probability $P_{ll}^{approx} = P_{acq}^{km} P_{pre}^{m}(i) P_{int}^{o}$ the prediction of the i-th time series element is accurate and with probability $P_{acq}^{km} P_{pre}^{m} P_{int}^{ol}$ the prediction of the complete time series is without any inaccurate context time series element.

Similar to the discussion above, for a prediction based on low-level context elements, with probability $(1 - P_{acq}^{k})$, an error occurs in one of the k context acquisition steps associated with a singular context source. With probability $(1 - P_{pre}(i))$ an error occurs in the context prediction step associated with one of the m dimensions of the low-level context time series. With Probability

$$P_{cor}^{pre} = (1 - P_{acq}^{k})(1 - P_{pre}(i)) \frac{1}{v_l - 1} \tag{5.6}$$

an error that occurred in the context acquisition step of one specific context source is corrected by one of the $v_l - 1$ possible errors in the context prediction step of time series element i. With probability

$$\left(P_{acq}^{k} P_{pre}(i) + P_{cor}^{pre}\right)^{m} P_{int}^{o} \tag{5.7}$$

the predicted low-level context element i is correct, even though an error may have occurred in the context acquisition and context prediction part, while no error occurred in the interpretation step. If we also consider errors in the context interpretation step, we obtain the probability $P_{ll}(i)$ that time series element i is accurately predicted as

$$\begin{aligned} P_{ll}(i) &= \left(P_{acq}^{k} P_{pre}(i) + P_{cor}^{pre}\right)^{m} P_{int}^{o} \\ &+ \left(1 - \left(P_{acq}^{k} P_{pre}(i) + P_{cor}^{pre}\right)^{m}\right) \frac{1 - P_{int}^{o}}{v_h^{o} - 1}. \end{aligned} \tag{5.8}$$

When considering the whole predicted time series T of length $|T| = n$ instead of single time series elements, the probability that the prediction

is without any inaccurate context time series element is

$$\begin{aligned} P_{ll} &= \left(P_{acq}^{k} P_{pre} + \left(1 - P_{acq}^{k}\right)(1 - P_{pre})\,\frac{1}{v_l^n - 1}\right)^m P_{int}^{o} \qquad (5.9) \\ &+ \left(1 - \left(P_{acq}^{k} P_{pre} + \left(1 - P_{acq}^{k}\right)(1 - P_{pre})\,\frac{1}{v_l^n - 1}\right)^m\right) \\ &\cdot \left(\frac{1 - P_{int}^{o}}{v_h^o - 1}\right)^n. \end{aligned}$$

Discussion

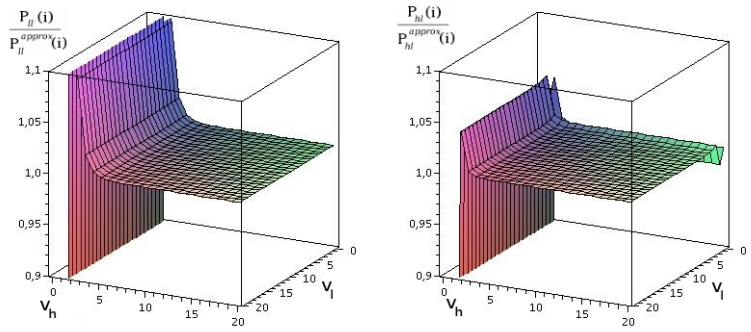
Having derived the context prediction accuracies for low-level and high-level context prediction schemes, we now discuss the impact of the context abstraction level on the context prediction accuracy. We explore this impact by a comparison of $P_{ll}(i)$ and $P_{hl}(i)$. These probabilities describe the case that one single high-level context element is predicted. It is clear that all findings that hold for $P_{ll}(i)$ and $P_{hl}(i)$ can be generalised to predicted context sequences of greater length. However, the formulae $P_{ll}(i)$ and $P_{hl}(i)$ are hard to grasp due to the multitude of variables involved that reappear in various parts of the terms. However, for two basic trends these formulae can be approximated by the simplified terms

$$P_{ll}^{approx}(i) = P_{acq}^{km} P_{pre}^{m}(i) P_{int}^{o} \qquad (5.10)$$

$$P_{hl}^{approx}(i) = P_{acq}^{km} P_{pre}^{o}(i) P_{int}^{ko}. \qquad (5.11)$$

In these formulae, the possibility to accidentally correct errors by a further error is not considered. This approximation is feasible for the following reasons. On the one hand, for $P_{acq} \to 1$, $P_{int} \to 1$ and $P_{pre}(i) \to 1$, the terms that describe the probabilities that errors are corrected by other errors are cancelled out. However, in this case the differences between high-level and low-level context prediction are only minor. We assume that this case is rather unrealistic as it implies that all error probabilities approach zero.

Another trend that leads to the same simplified terms is dependent on v_l and v_h. For $v_l \to \infty$ and $v_h \to \infty$ the high-level and low-level prediction accuracies can be approximated by $P_{ll}^{approx}(i)$ and $P_{hl}^{approx}(i)$. Figure 5.2 shows $\frac{P_{ll}(i)}{P_{ll}^{approx}(i)}$ and $\frac{P_{hl}(i)}{P_{hl}^{approx}(i)}$ respectively for $k = m = o = 5, P_{acq} = 0.99, P_{pre}(i) = 0.9, P_{int} = 0.9$.



(a) $\frac{P_{ll}(i)}{P_{ll}^{approx}(i)}$ (b) $\frac{P_{hl}(i)}{P_{hl}^{approx}(i)}$

Figure 5.2: Comparison of approximated to exact probability of prediction errors for $k = m = o = 5$ and $P_{acq} = 0.99, P_{int} = P_{pre}(i) = 0.9$.

These parameters are chosen to represent settings in typical scenarios. First of all, we assume the acquisition error to be rather small since acquisition is basically a transformation of sampled values to a common representation. Accordingly, the acquisition error is chosen to be as small as 0.01. For context prediction and context interpretation we expect higher error probabilities. In scenarios with well understood properties, we assume that error probabilities better than 0.1 are reasonable. For context prediction this reflects results obtained for example in financial forecasting when, for instance, an ARMA approach is applied.

For the other parameters we differentiate between small, medium and large scale scenarios and set the parameters as 5, 20 or 40 accordingly. While in small scenarios a number of five different sensors, for example, that are utilised in parallel are typical, a scenario with up to 40 different context sources in parallel can be considered as large indeed.

From figure 5.2 we see that for sufficiently large numbers of high-level or low-level context values v_l and v_h the approximation functions sufficiently describe $P_{ll}(i)$ and $P_{hl}(i)$.

In typical scenarios, the number of values a high-level or low-level context may take is easily above 10 or 20. This is especially true for low-level context values. Consider, for example, a temperature sensor. The sensor readings might, for instance, be mapped to integer values in the range $[0, 30]$, which corresponds to the case that 30 distinct low-level context values are possible for this one context type.

For high-level contexts, these values might be mapped to values as, for example, 'cold' or 'warm'. However, in ambiguous scenarios, also in the high-level domain, further context values become necessary. Examples are room temperature, sleeping-temperature, outdoor-temperature-summer, outdoor-temperature-winter as well as distinctions between various rooms since people typically have different temperature-preferences for bathroom, livingroom or kitchen.

Summarising we can say that the number of values possible for v_l and v_h in realistic scenarios are typically quite large.

For sufficiently large values of v_l and v_h, observations made for $P_{ll}^{approx}(i)$ and $P_{hl}^{approx}(i)$ are therefore also valid for $P_{ll}(i)$ and $P_{hl}(i)$. We therefore first discuss $P_{ll}^{approx}(i)$ and $P_{hl}^{approx}(i)$ before considering the more exact formulae $P_{ll}(i)$ and $P_{hl}(i)$. First of all, we observe that the influence of acquisition errors is equal for high-level and low-level context prediction schemes, since the factor P_{acq}^{km} appears in both formulae.

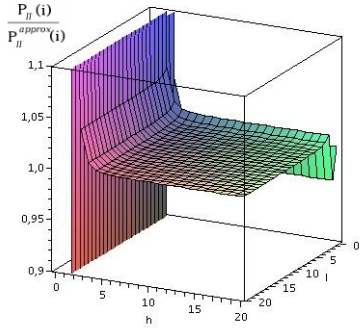
The fraction of these probabilities yields

$$\frac{P_{hl}^{approx}(i)}{P_{ll}^{approx}(i)} = P_{int}^{k} P_{pre}^{o-m}(i). \tag{5.12}$$

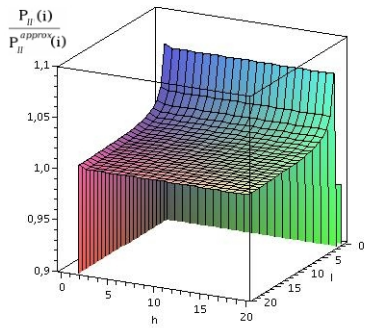
Clearly, this term is smaller than 1 for all configurations other than $P_{int} = P_{pre}(i) = 1$. Consequently, for sufficiently large values of v_l and v_h, context prediction based on low-level context elements is superior to context prediction based on high-level context elements.

However, as figure 5.3 illustrates, the values required for v_l and v_h in order for the approximation to hold, increase when $P_{pre}(i)$ and P_{int} decrease or the values of k, m or o increase. We therefore also study $P_{hl}(i)$ and $P_{ll}(i)$ for small values of v_l and v_h.

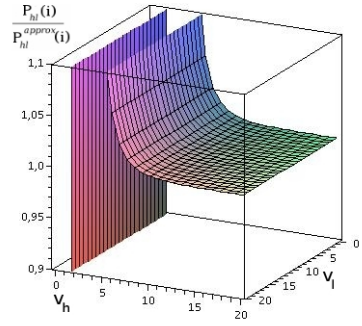
We identify differences between $P_{hl}(i)$ and $P_{ll}(i)$ according to the values of the input parameters used. The basic structure of both formulae is similar. The most interesting differences are related to interdependencies between P_{int} and $P_{pre}(i)$ as well as to the value m that describes the low-level dimensionality.



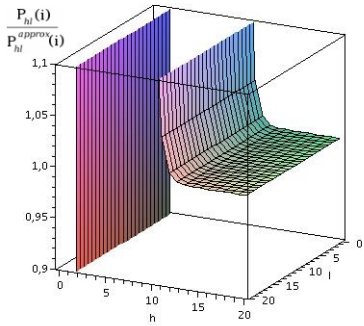
(a) $k = m = o = 5$ and $P_{acq} = 0.99, P_{int} = P_{pre}(i) = 0.8$



(b) $k = m = o = 20$ and $P_{acq} = 0.99, P_{int} = P_{pre}(i) = 0.9$



(c) $k = m = o = 5$ and $P_{acq} = 0.99, P_{int} = P_{pre}(i) = 0.8$



(d) $k = m = o = 20$ and $P_{acq} = 0.99, P_{int} = P_{pre}(i) = 0.9$

Figure 5.3: Comparison of $P_{ll}(i)$ and $P_{ll}^{approx}(i)$.

In figure 5.4 the probabilities that a prediction based on high-level context elements has no inaccurately predicted context elements are depicted for several values of $P_{pre}(i)$ and P_{int}. For ease of presentation, we show the results only for $P_{pre}(i), P_{int} \geq 0.96$. For smaller values of $P_{pre}(i)$ and P_{int} the general properties shown are identical. From the left picture to the right picture the values of the variables v_h, k, v_l, m, o are set to $5, 20$ and 40, while P_{acq} is set to 0.999. Generally, we observe that the probability for a correct prediction decreases with increasing value for v_h, k, v_l, m and o as could be expected.

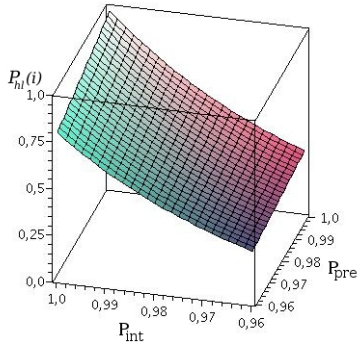
For low-level context prediction this general property is, of course, unchanged (cf. Figure 5.5). However, the degradation is not as harsh as for high-level context prediction. We therefore conclude that the low-level prediction scheme is better capable of dealing with this configuration of the input parameters v_h, k, v_l, m and o.

We illustrate the predominance of the low-level context prediction scheme above the high-level context prediction scheme by dividing the low-level probability $P_{ll}(i)$ by the high-level probability $P_{hl}(i)$: $\frac{P_{ll}(i)}{P_{hl}(i)}$. For $P_{acq} = 0.999, k = v_l = v_h = m = o = 5$ the result of this fraction is depicted in figure 5.6 for several values of $P_{pre}(i)$ and P_{int}. In these figures at all points below 1.0 the high-level context prediction scheme is superior, while at all points above 1.0 the low-level context prediction scheme performs better. In order to obtain an impression for how many configurations the low-level context prediction scheme is superior to the high-level context prediction scheme, we display this fraction only for $0 < \frac{P_{ll}(i)}{P_{hl}(i)} \leq 1$, which results in all points at which the high-level context prediction scheme is not inferior.

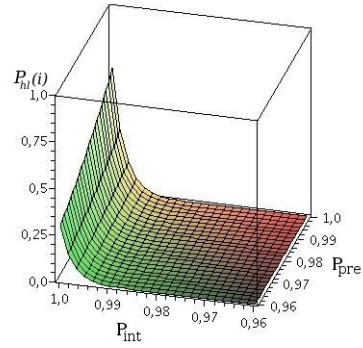
These points are depicted in figure 5.7 for arbitrary values of P_{int} and $P_{pre}(i)$ and $k = v_l = v_h = m = o \in [5, 20, 40]$. We observe that low-level context prediction has the lower probability of error for all but low values of $P_{pre}(i)$. The number of points where the low-level context prediction is superior to the high-level context prediction increases for higher values of v_h, k, v_l, m and o.

Observe that the high-level context prediction scheme is only superior for values of $P_{pre}(i)$ that are below 0.25. We argue that these low values for the prediction process are not acceptable for any utilisation of context prediction in real world scenarios.

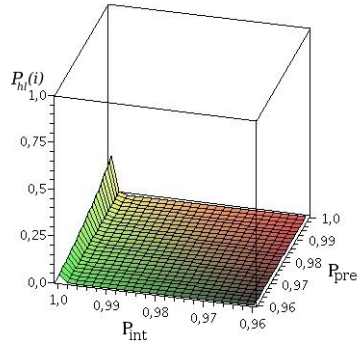
Impact of the acquisition accuracy: When P_{acq} is decreased, the overall probability for a correct prediction decreases as expected for high-



(a) $P_{acq} = 0.999$ and $k = m = o = v_l = v_h = 5$

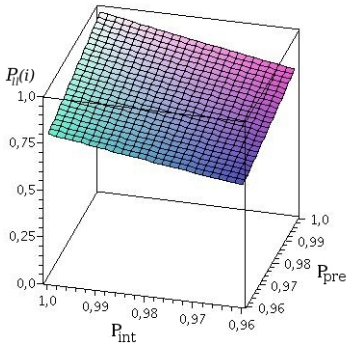


(b) $P_{acq} = 0.999$ and $k = m = o = v_l = v_h = 20$

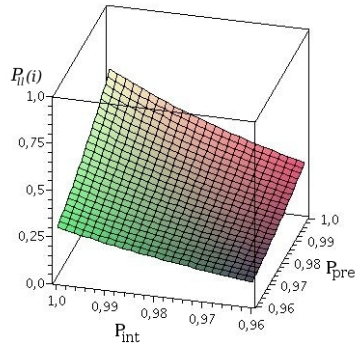


(c) $P_{acq} = 0.999$ and $k = m = o = v_l = v_h = 40$

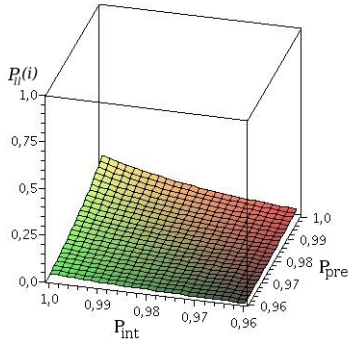
Figure 5.4: High-level context prediction probability that no error occurs in the context prediction process of the i-th high-level context time series element.



(a) $P_{acq} = 0.999$ and $k = m = o = v_l = v_h = 5$

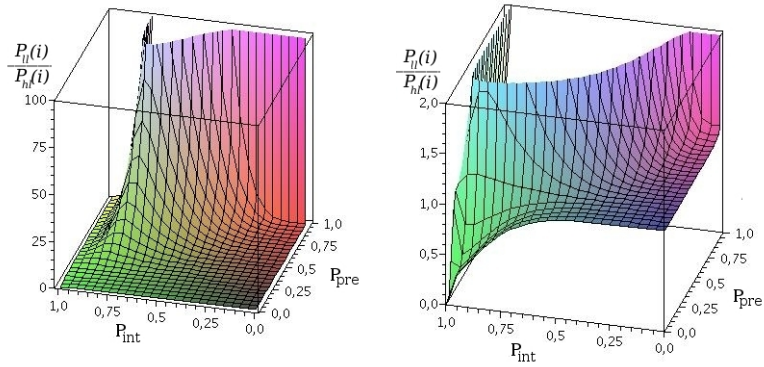


(b) $P_{acq} = 0.999$ and $k = m = o = v_l = v_h = 20$



(c) $P_{acq} = 0.999$ and $k = m = o = v_l = v_h = 40$

Figure 5.5: Low-level context prediction probability that no error occurs in the context prediction process of the i-th high-level context time series element.



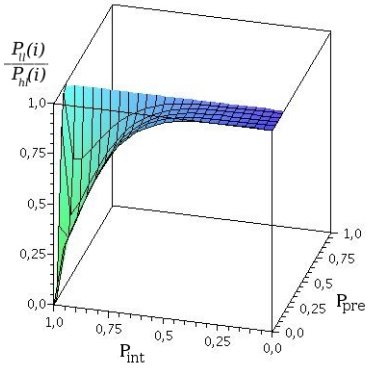
(a) $\frac{P_{ll}(i)}{P_{hl}(i)}$ in the range of 0 to 100.

(b) $\frac{P_{ll}(i)}{P_{hl}(i)}$ in the range of 0 to 2.

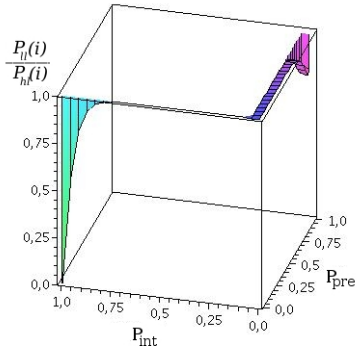
Figure 5.6: Comparison of the low-level and high-level context prediction schemes.

level and low-level context prediction schemes (cf. figures 5.8 and 5.9). In the figure v_h, k, v_l, m and o are set to 5 while P_{acq} is subsequently decreased from 0.999 to 0.95 to 0.9. We observe that the impact is serious for high-level and low-level context prediction schemes. This is especially true since the maximum probability that is reachable for an accurate prediction is seriously lowered when P_{acq} is lowered. The error probability of the acquisition process is therefore a most critical input to the overall prediction process regardless of the exact prediction scheme utilised. However, the slope of the probability plane and consequently the probability of an error is still higher in case of the high-level context prediction scheme for all values of P_{acq}.

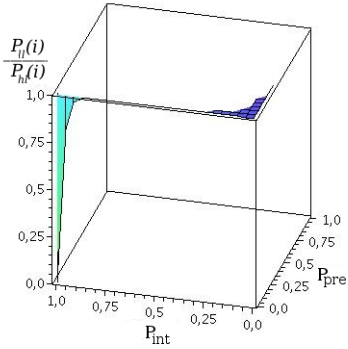
Impact of the number of context values: Next we study the influence of the parameter v_l which describes the number of possible low-level context values. Figure 5.10 and figure 5.11 depict the probability that no error occurs in the prediction process for high-level and low-level prediction schemes respectively. In these figures P_{acq} is set to 0.999 while v_h, k, m and o are set to 5 and v_l is varied between $5, 20$ and 40. As can be observed from the figures, neither for low-level nor for high-level context prediction is the overall probability for an accurate prediction decreased significantly.



(a) $v_h, k, v_l, m, o = 5,\ P_{acq} = 0.999$.

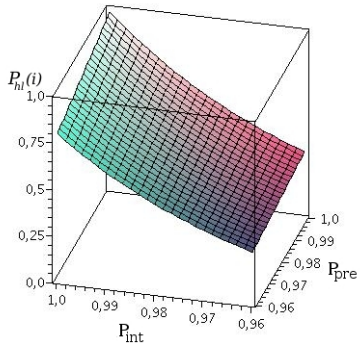


(b) $v_h, k, v_l, m, o = 5,\ P_{acq} = 0.95$.

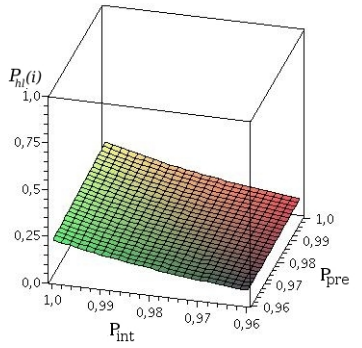


(c) $v_h, k, v_l, m, o = 5,\ P_{acq} = 0.9$.

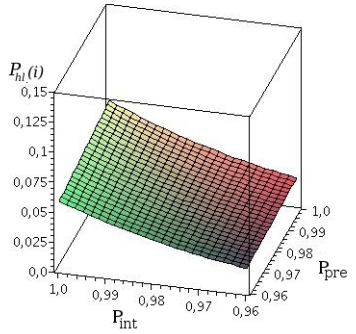
Figure 5.7: Regions in the probability space where the high-level context prediction scheme outperforms the low-level context prediction scheme.



(a) P_{hl} for $P_{acq} = 0.999$ and $k = m = o = v_l = v_h = 5$

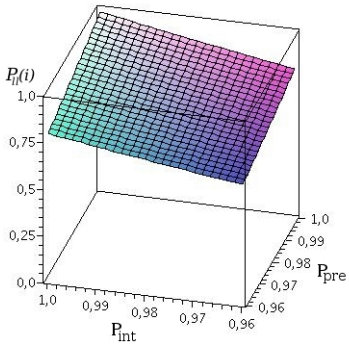


(b) P_{hl} for $P_{acq} = 0.95$ and $k = m = o = v_l = v_h = 5$

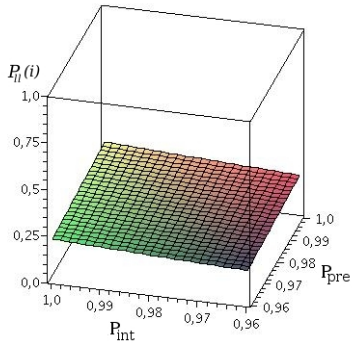


(c) P_{hl} for $P_{acq} = 0.9$ and $k = m = o = v_l = v_h = 5$

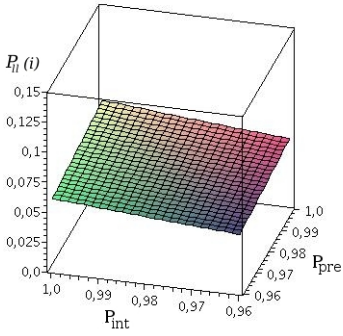
Figure 5.8: Probability planes describing the probability that the high-level context prediction scheme is without error.



(a) P_{ll} for $P_{acq} = 0.999$ and $k = m = o = v_l = v_h = 5$



(b) P_{ll} for $P_{acq} = 0.95$ and $k = m = o = v_l = v_h = 5$



(c) P_{ll} for $P_{acq} = 0.9$ and $k = m = o = v_l = v_h = 5$

Figure 5.9: Probability planes describing the probability that the low-level context prediction scheme is without error.

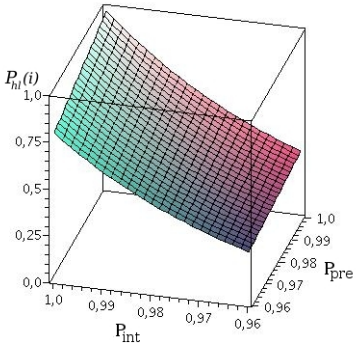
Hence, the influence of the number of possible low-level context types can be considered as marginally at most. However, the probability plane for low-level context prediction schemes lies above the probability plane for high-level context prediction schemes. Low-level context prediction schemes are therefore predominant in this case also. As can be observed from figure 5.12, figure 5.13 and figure 5.14, this property is similar for the value v_h of different high-level context time series.

Impact of the low-level time series dimension: For the parameter m that describes the number of raw data values utilised during the prediction process, as well as the dimension of the low-level context time series, we observe that the context prediction accuracy decreases with increasing number of context sources for high-level and low-level context prediction schemes. However, the impact on the high-level prediction scheme is higher for small values of m (cf. figure 5.15 and figure 5.16). In the figures, P_{acq} is set to 0.999 and v_h, k, v_l and o are set to 5 while m is varied from 5 to 20 to 40. While the higher probability in case of low-level context prediction is evident from the figures for $m = 5$, for higher values of m, generally the low-level context prediction performs better for configurations with higher values of $P_{pre}(i)$, whereas for high-level context prediction the accuracy is better for higher values of P_{int}. In figure 5.17 the fraction $\frac{P_{ll}(i)}{P_{hl}(i)}$ is depicted for $v_h = k = v_l = o = 5$ and $P_{acq} = 0,999$ while m takes the values $5, 20$ and 40 respectively.

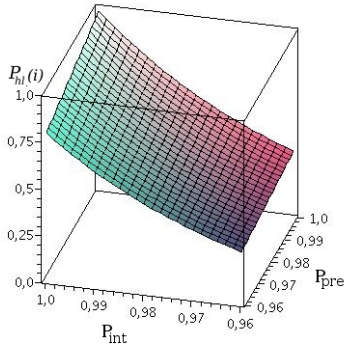
In the figure we observe from the middle and right pictures for $m = 20$ and $m = 40$ that the low-level context prediction scheme is roughly advantaged for $P_{pre}(i) > P_{int}$, while the high-level context prediction scheme has the higher accuracy for $P_{int} > P_{pre}(i)$. Therefore, for high values of m the fraction of P_{int} to $P_{pre}(i)$ determines which context prediction scheme is to be utilised for the context prediction task.

Impact of the high-level time series dimension: The value o that describes the number of parallel high-level context time series has significant impact on the context prediction accuracy. In figure 5.18 and figure 5.19 we observe that the impact of this value is more significant for high-level context prediction.

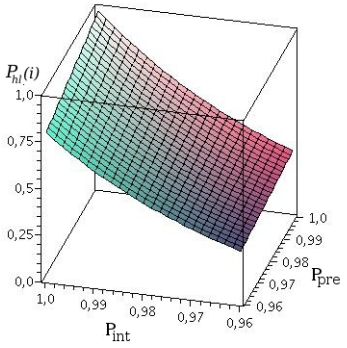
In figure 5.20 we observe that for various values of P_{int} and $P_{pre}(i)$ the low-level context prediction is more accurate for most of the sampled points. Moreover, the high-level context prediction scheme is only for less relevant low values of P_{int} or $P_{pre}(i)$ more accurate than the low-level context prediction scheme.



(a) $P_{acq} = 0.999, v_h = v_l = k = m = o = 5$.

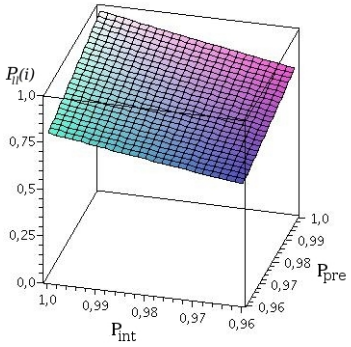


(b) $P_{acq} = 0.999, v_l = 20, v_h = k = m = o = 5$.

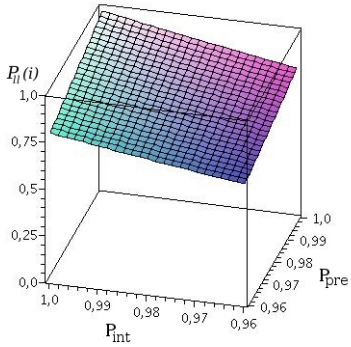


(c) $P_{acq} = 0.999, v_l = 40, v_h = k = m = o = 5$.

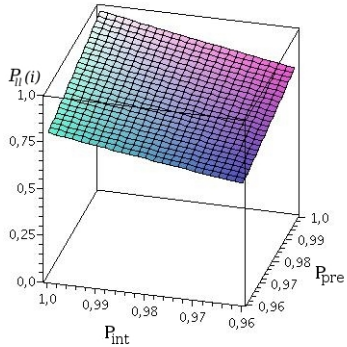
Figure 5.10: Probability planes for high-level context prediction when the number of low-level context types is varied.



(a) $P_{acq} = 0.999, v_h = v_l = k = m = o = 5$.

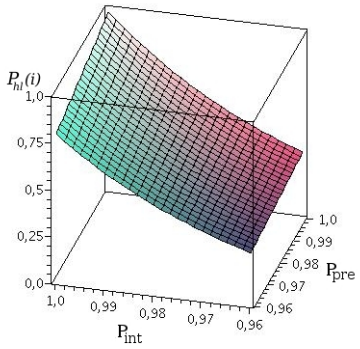


(b) $P_{acq} = 0.999, v_l = 20, v_h = k = m = o = 5$.

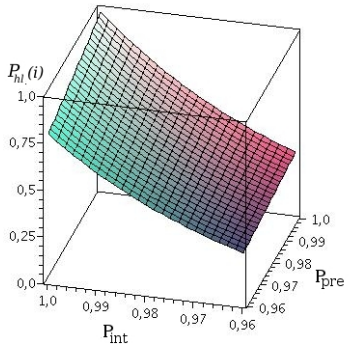


(c) $P_{acq} = 0.999, v_l = 40, v_h = k = m = o = 5$.

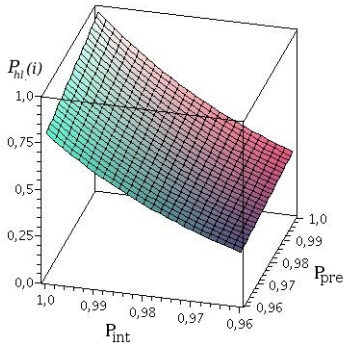
Figure 5.11: Probability planes for low-level context prediction when the number of low-level context types is varied.



(a) $P_{acq} = 0.999, v_l = v_h = k = m = o = 5$.

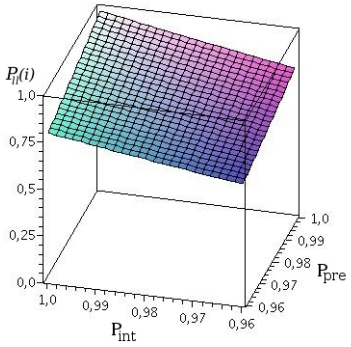


(b) $P_{acq} = 0.999, v_h = 20, v_l = k = m = o = 5$.

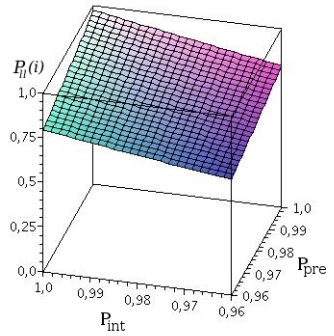


(c) $P_{acq} = 0.999, v_h = 40, v_l = k = m = o = 5$.

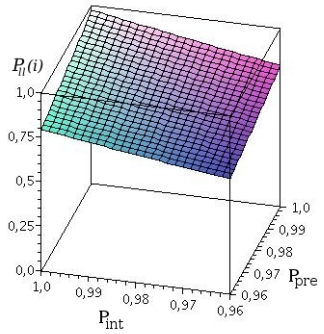
Figure 5.12: Probability planes for high-level context prediction when the number of high-level context types is varied.



(a) $P_{acq} = 0.999, v_l = v_h = k = m = o = 5$.

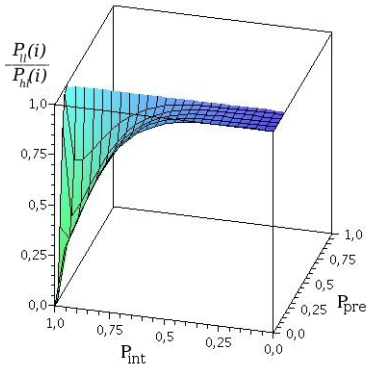


(b) $P_{acq} = 0.999, v_h = 20, v_l = k = m = o = 5$.

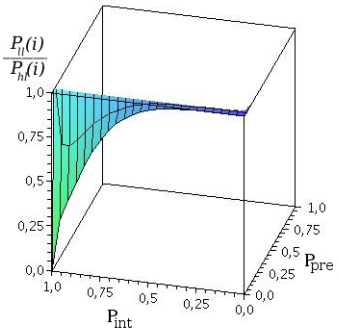


(c) $P_{acq} = 0.999, v_h = 40, v_l = k = m = o = 5$.

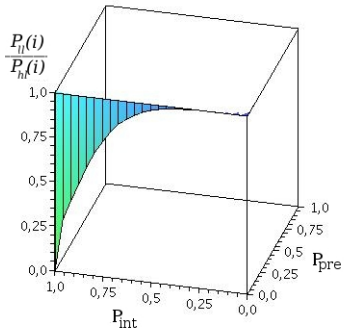
Figure 5.13: Probability planes for low-level context prediction when the number of high-level context types is varied.



(a) $\frac{P_{ll}(i)}{P_{hl}(i)}$ for $P_{acq} = 0.999, v_l = k = m = o = 5$ and $v_h = 5$

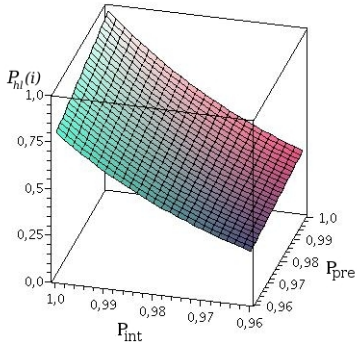


(b) $\frac{P_{ll}(i)}{P_{hl}(i)}$ for $P_{acq} = 0.999, v_l = k = m = o = 5$ and $v_h = 20$

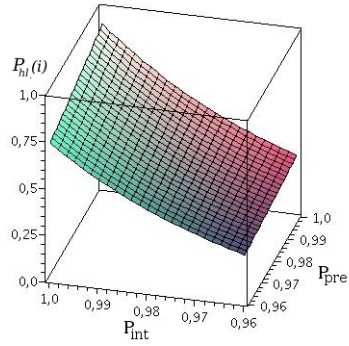


(c) $\frac{P_{ll}(i)}{P_{hl}(i)}$ for $P_{acq} = 0.999, v_l = k = m = o = 5$ and $v_h = 40$

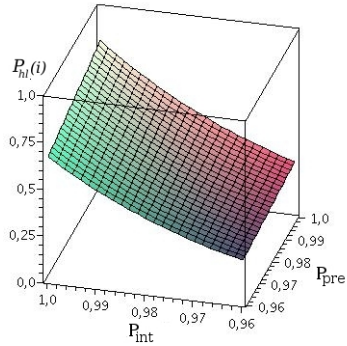
Figure 5.14: Comparison between low-level and high-level context prediction schemes for varying number of high-level context types.



(a) $P_{acq} = 0.999, v_l = v_h = k = m = o = 5$.

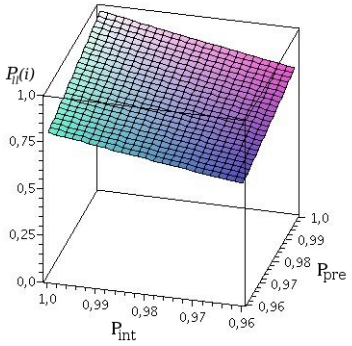


(b) $P_{acq} = 0.999, v_l = v_h = k = o = 5, m = 20$.

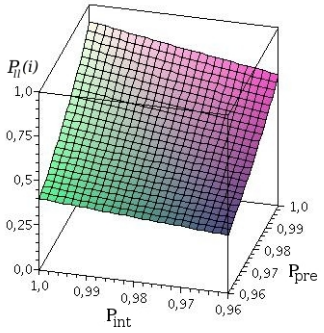


(c) $P_{acq} = 0.999, v_l = v_h = k = o = 5, m = 40$.

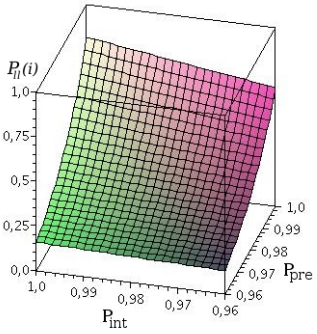
Figure 5.15: Probability planes for high-level context prediction when the dimension of the low-level context time series.



(a) $P_{acq} = 0.999, v_l = v_h = k = m = o = 5$.

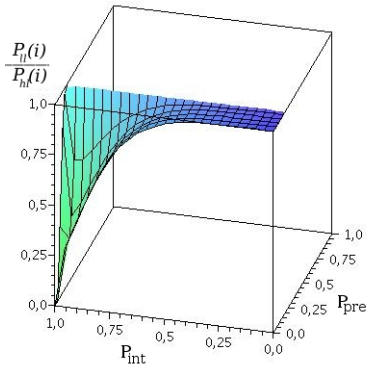


(b) $P_{acq} = 0.999, v_l = v_h = k = o = 5, m = 20$.

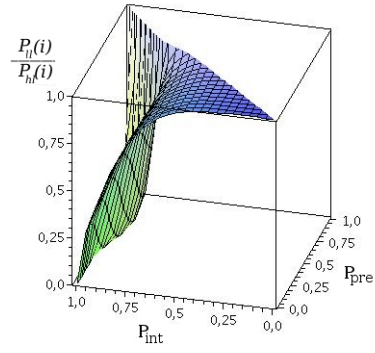


(c) $P_{acq} = 0.999, v_l = v_h = k = o = 5, m = 40$.

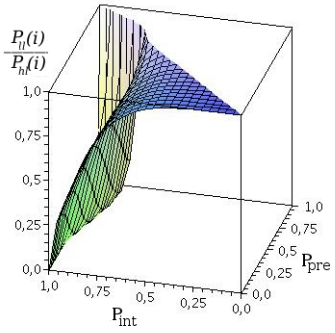
Figure 5.16: Probability planes for low-level context prediction when the dimension of the low-level context time series is varied.



(a) $\frac{P_{ll}(i)}{P_{hl}(i)}$ for $P_{acq} = 0.999, v_l = v_h = k = o = m = 5$

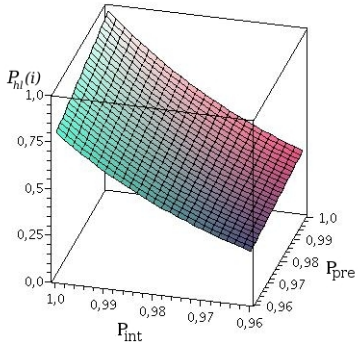


(b) $\frac{P_{ll}(i)}{P_{hl}(i)}$ for $P_{acq} = 0.999, v_l = v_h = k = o = 5, m = 20$

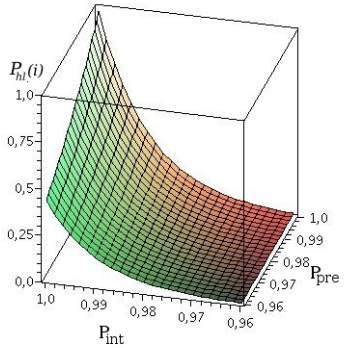


(c) $\frac{P_{ll}(i)}{P_{hl}(i)}$ for $P_{acq} = 0.999, v_l = v_h = k = o = 5, m = 40$

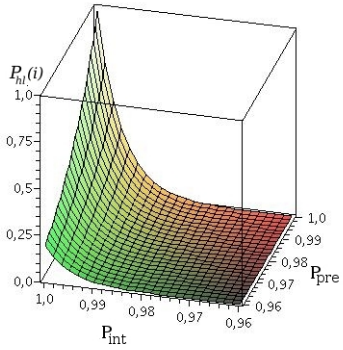
Figure 5.17: Comparison between low-level and high-level context prediction schemes for a varying dimension of the low-level context time series.



(a) $P_{acq} = 0.999, v_l = v_h = k = m = o = 5$.

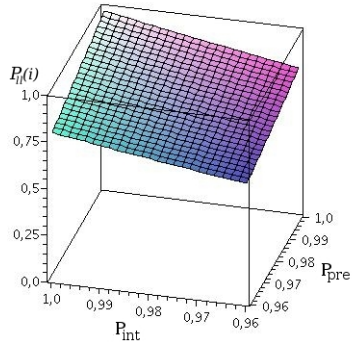


(b) $P_{acq} = 0.999, v_l = v_h = k = m = 5, 0 = 20$.

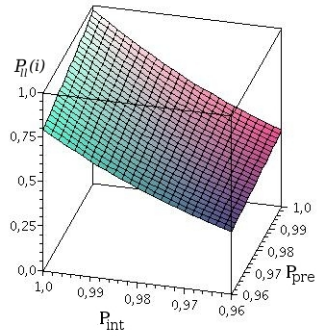


(c) $P_{acq} = 0.999, v_l = v_h = k = m = 5, o = 40$.

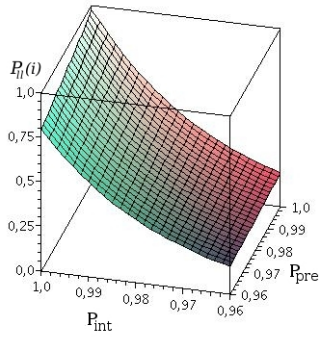
Figure 5.18: Probability planes for high-level context prediction when the dimension of the high-level context time series is varied.



(a) $P_{acq} = 0.999, v_l = v_h = k = m = o = 5$.

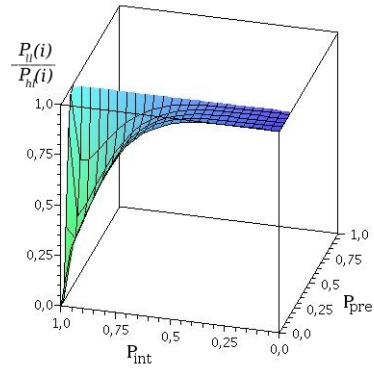


(b) $P_{acq} = 0.999, v_l = v_h = k = m = 5, 0 = 20$.

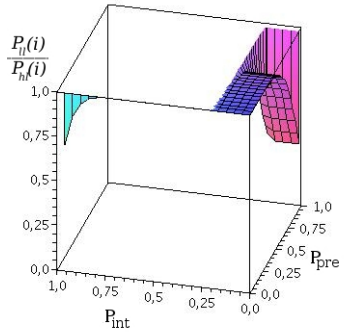


(c) $P_{acq} = 0.999, v_l = v_h = k = m = 5, o = 40$.

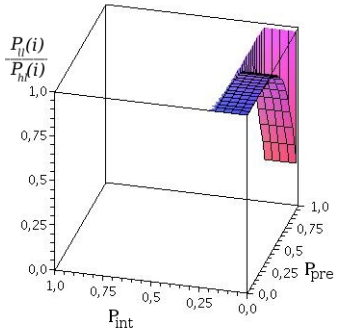
Figure 5.19: Probability planes for low-level context prediction when the dimension of the high-level context time series is varied.



(a) $\frac{P_{ll}(i)}{P_{hl}(i)}$ for $P_{acq} = 0.999, v_l = v_h = k = m = o = 5$



(b) $\frac{P_{ll}(i)}{P_{hl}(i)}$ for $P_{acq} = 0.999, v_l = v_h = k = m = 5, o = 20$



(c) $\frac{P_{ll}(i)}{P_{hl}(i)}$ for $P_{acq} = 0.999, v_l = v_h = k = m = 5, o = 40$

Figure 5.20: Comparison between low-level and high-level context prediction schemes for a varying dimension of the high-level context time series.

Impact of the context prediction horizon: For the value k that describes the context prediction horizon, we again observe that the high-level context prediction scheme has the higher probability of error (cf. figure 5.21, figure 5.22 and figure 5.23). This property intensifies with an increasing size of the context history.

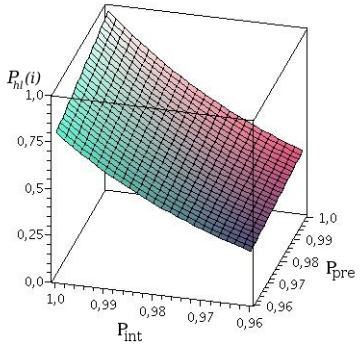
Summary

We have observed that the accuracy of the high-level context prediction scheme is lower than the accuracy of the low-level context prediction scheme for sufficiently large values of v_l and v_h that describe the number of different values for high-level and low-level context elements.

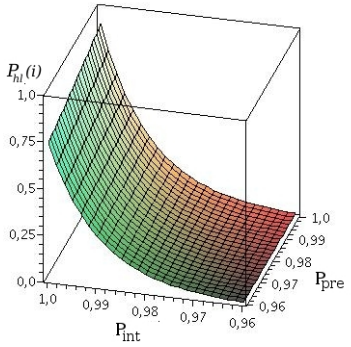
For lower values of v_l and v_h this property is marginally altered in favour of the high-level context prediction scheme. However, we also observed that the high-level context prediction scheme becomes superior in many cases to the low-level context prediction scheme when the number of input dimensions increases. Figure 5.17 details that with input dimensions of 20 or 40 the fraction of interpretation to prediction accuracy determines the prediction scheme of choice. In such scenarios with a great number of different context sources, architectures with accurate interpretation modules but less accurate prediction modules should favour high-level prediction schemes, while it is complementary when the prediction accuracy is good and the interpretation accuracy not.

In realistic settings, scenarios with 20 or 40 different context sources are currently seldom. However, this might easily change with context awareness being introduced in an increasing number of devices. When the environment is temporarily fluctuating at a fast pace we can easily imagine that context prediction is less accurate than context interpretation. In such a scenario, context prediction based on high-level context time series is likely to be more accurate than context prediction based on low-level context time series.

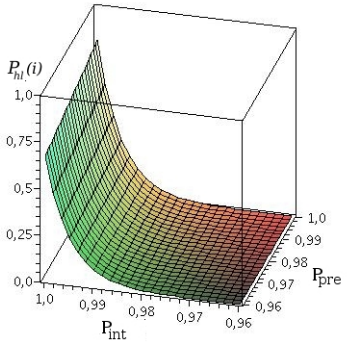
As an example scenario that might fulfil the requirements described above, consider for example the modelling of context of some kind of traffic system as, for instance, a railroad system or a car traffic system in a city. One might be interested in the overall or area specific traffic situation. We expect a high number of different sensors at various points in the traffic system in this scenario. Potential sensors might, for example, measure the congestion at strategic crossroads, the air pollution in rural areas as well



(a) $P_{acq} = 0.999, v_l = v_h = m = o = k = 5$.

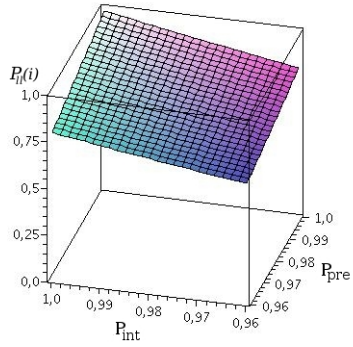


(b) $P_{acq} = 0.999, v_l = v_h = m = o = 5, k = 20$.

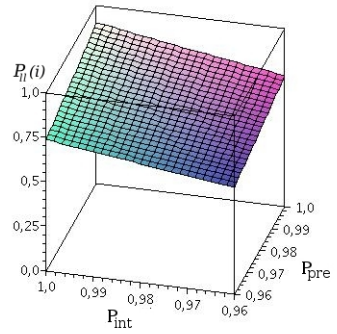


(c) $P_{acq} = 0.999, v_l = v_h = m = o = 5, k = 40$.

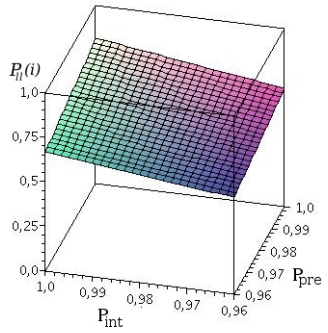
Figure 5.21: Probability planes for high-level context prediction when the low-level and high-level context history size is varied.



(a) $P_{acq} = 0.999, v_l = v_h = m = o = k = 5$.

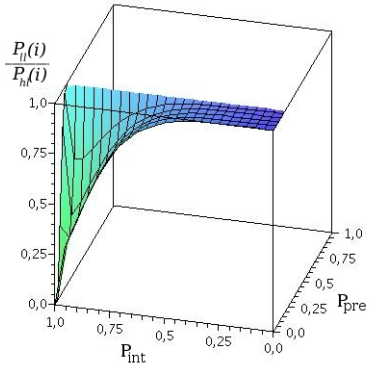


(b) $P_{acq} = 0.999, v_l = v_h = m = o = 5, k = 20$.

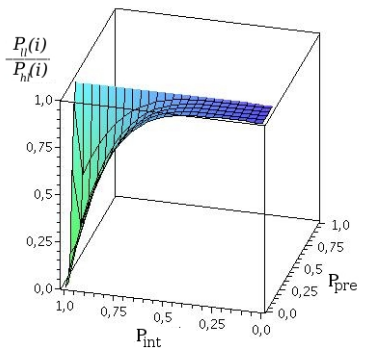


(c) $P_{acq} = 0.999, v_l = v_h = m = o = 5, k = 40$.

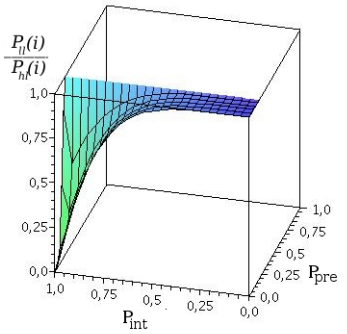
Figure 5.22: Probability planes for low-level context prediction when the low-level and high-level context history size is varied.



(a) $\frac{P_{ll}(i)}{P_{hl}(i)}$ for $P_{acq} = 0.999, v_l = v_h = m = o = k = 5$.



(b) $\frac{P_{ll}(i)}{P_{hl}(i)}$ for $P_{acq} = 0.999, v_l = v_h = m = o = 5, k = 20$.



(c) $\frac{P_{ll}(i)}{P_{hl}(i)}$ for $P_{acq} = 0.999, v_l = v_h = m = o = 5, k = 40$.

Figure 5.23: Comparison between low-level and high-level context prediction schemes for a varying size of the context history.

as the traffic flow through the system or the types of vehicles that account for the traffic.

Summarising, for increasing number of raw data values that are utilised for the prediction schemes, the fraction of interpretation probability to prediction probability becomes more important. As a rule of thumb, high-level context prediction schemes increase in predominance when the fraction $\frac{P_{int}}{P_{pre}(i)}$ increases. However, for all other parameters studied, the low-level context prediction scheme is predominant.

We have observed therefore that the high-level context prediction scheme has the higher probability to compute inaccurate context predictions even for environments where the context time series dimensionality as well as the context history size are small. With increasing size of the environment or scenario where the prediction scheme is applied, this property intensifies for nearly all relevant parameters.

Further advantages of the high-level context prediction scheme could only be observed for significantly low values of P_{int} or $P_{pre}(i)$ that are not feasible in realistic scenarios since the probability of error is far above $\frac{1}{2}$.

The number of high-level and low-level context types have no significant impact on both context prediction schemes.

Generally, the context prediction accuracy is highly connected to the context acquisition accuracy. The impact of P_{acq} is higher than the impact of P_{int} and $P_{pre}(i)$ together. Consequently, the main attention of the application designer should focus the context acquisition procedure. Furthermore, designers of context prediction architectures have to consider the ratio of prediction to interpretation accuracy, as well as the dimension of the context history in order to achieve a system with maximum accuracy. The number of context types available, however, has minor influence on the context prediction accuracy.

Probability estimation for specific scenarios For the obtained results to hold, several assumptions have to be taken beforehand. As we have stated above, some of the assumptions that have been made in order to provide a most comprehensive general discussion might not apply to specific application scenarios.

Of particular significance, the fixed number of context values for context types of one data abstraction level and the fixed probability for every application of one processing step, as well as the uniform distribution of errors, might differ in distinct application scenarios.

However, the formulae developed might provide a decent starting point for the analysis of specific application scenarios.

In scenarios where the number of values for contexts of one context abstraction level is not constant, additional variables $v_h(\mu)$ and $v_l(\nu)$ have to be introduced to cover all possible values.

The same is generally true for differing probabilities for every application of a processing step. In case of other error distributions, the model of the uniform distribution in the formula has to be exchanged by alternative probability distributions.

5.1.2 Impact of the size of the prediction search space

Apart from the accuracies of the context processing steps involved in the context prediction process, the context abstraction level also influences the context prediction accuracy even when no errors occur in any of the processing steps. Due to a higher context abstraction level, some information is hidden from the context prediction algorithm.

As discussed in section 3.2.5, some information contained in a low-level context is lost when transformed to a high-level context.

One example is the measurement of a user trajectory based on GPS-coordinates (low-level contexts) as opposed to labelled context locations (high-level contexts). A summary of this setting is depicted in figure 5.24. The observed time series of low-level GPS-coordinates is depicted on the left hand side of the map while the time series of labelled high-level contexts can be found to the right of the map. From top to bottom the low-level and high-level contexts of a person moving along the dashed line in the figure are depicted. The associated low-level contexts indicate a general trend while the high-level contexts mask this trend to some extend due to the higher level of abstraction.

As stated in section 3.2.4, the context prediction problem is a search problem. Due to the differing context abstraction level, the basic parameters of the search problem also differ from high-level to low-level context prediction. This means that for high-level and low-level context prediction the search space may differ even though the sampled raw context data is identical. The reason for this is the possible higher abstraction level for high-level contexts as we have discussed above. We discuss the influence of the different search spaces that are the basis for high-level and low-level context prediction methods.

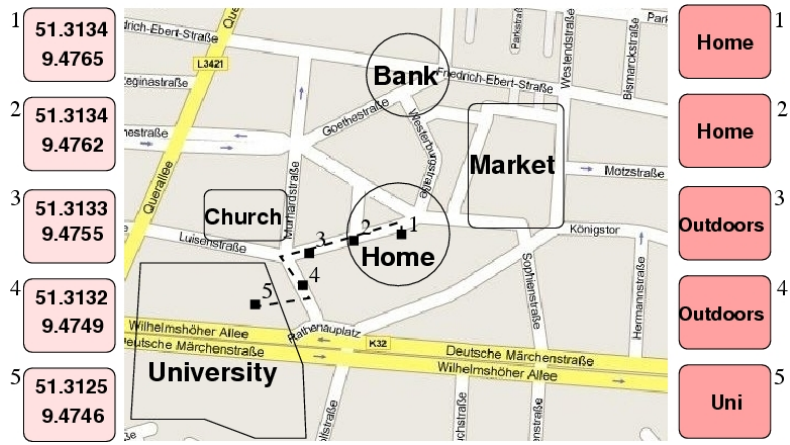


Figure 5.24: High-level and low-level trajectory of a mobile user.

Let $k \in \mathbb{N}$ be the length of the context history for high-level and low-level context prediction schemes. The search space S_l (S_h) of a low-level (high-level) context prediction algorithm is composed of all possible K_l (K_h) context patterns of length k. For this purpose, assume that $K_l = (v_l^m)^k$ be the maximum number of different low-level context patterns of length k and $K_h = (v_h^o)^k$ be the maximum number of different high-level context patterns. Therefore, K_l (K_h) is also the number of context time series that may be unambiguously represented in the search space S_l (S_h). Consequently, for $K_l, K_h < \infty$ the capacity of the search spaces S_l and S_h is finite.

In principle, in case of real valued outputs of context sources, the number of points in the search space may be infinite. However, in a realistic scenario, any context source that measures a physical quantity has at least maximum and minimum values it can produce, and only a finite number of possible values in between.

Due to natural measurement errors all raw data values are already expected to be error prone. We therefore define an error threshold that may vary with every context source attached to a context prediction architecture. A measured value is then considered correct, when the measurement

error does not violate the error threshold.

Let $a \in \mathbb{N}$ and $b \in \mathbb{N}$ be the maximum and minimum output values of a context source and $\delta \in \mathbb{R}$ be the error threshold of any one context source. We divide the output range of a context source into $\frac{|a-b|}{2\delta}$ sections of the same width 2δ. If a measured value lies inside section $\lambda, \lambda \in \{1, \ldots, \frac{|a-b|}{2\delta}\}$ the output is mapped to $\delta(2\lambda - 1)$ (the value representing the middle of this section).

If high-level contexts are derived from low-level contexts by grouping several values of various low-level contexts together, the granularity and possibly also the dimension of the high-level search space becomes lower than the dimensionality and granularity of the low-level search space ($|S_h| \leq |S_l|$). In this case, there must exist different low-level time series that are mapped to the same high-level time series. Therefore, the information content that can be expressed by a low-level time series of length k is higher than the information contained in a high-level time series of the same length. Consequently, when the prediction is based on low-level context elements instead of high-level context elements, a prediction of the same prediction horizon may be possible with reduced context history size.

Robustness to acquisition errors

An argument occasionally raised in favour of the high-level prediction approach is that errors that occur in the acquisition step might be consciously corrected with the world knowledge available in the interpretation step. Following this argumentation, high-level contexts become less error prone than low-level contexts. This would counter one of our arguments why low-level context prediction has a lower probability of error.

As we have discussed in section 2.2.4, the context interpretation step does not include a possibility for correction of erroneous contexts. We derive in this section that a correction of the acquired contexts is only possible on low-level context elements.

A correction of erroneously acquired context elements is an additional process that is not related to high-level or low-level context prediction schemes but may be utilised by both schemes alike.

In the following discussion we assume a process π that creates the environmental variables which are measured by the context sources attached to a context prediction architecture. We first consider minor measurement

errors.

Definition 5.1.1 : Minor measurement errors

Let d_i be the value of a measurable entity and $\upsilon_i = d_i + \varepsilon_i$ the measured value. Further, let $a, b \in \mathbb{R}, a < b$ be the lower and upper bound of the output range of the corresponding context source and $\delta \in \mathbb{R}$ be the error threshold. ε_i is a minor measurement error when the equation

$$\left\lceil \frac{d_i}{\frac{a-b}{2\delta}} \right\rceil = \left\lceil \frac{\upsilon_i}{\frac{a-b}{2\delta}} \right\rceil \tag{5.13}$$

holds.

In the following discussion we distinguish between correct contexts or time series and measured contexts or context time series. Correct contexts possess context values without any measurement error and a correct context time series is exclusively composed of correct contexts. Since a minor measurement error is smaller than the granularity we agreed on, it has no effect on either prediction method and might therefore be ignored.

Larger measurement errors A large measurement error is an error that exceeds the error threshold. We discuss the question whether this derivation leads to a different interpretation of the low-level context to the high-level context case as well as how and if it can be detected and corrected.

It is important for the following discussion to distinguish between the knowledge required for the interpretation step and the interpretation step itself. As we have seen in section 2.2.4, the knowledge utilised in the interpretation step enables the transformation from low-level to high-level contexts. We can envision this knowledge as a set of inequalities that describe a region in the multi-dimensional search space which represents all measured low-level context values (cf. section 2.2.4).

This knowledge is not bound to the interpretation step and can also be utilised by other modules as, for example, an error correction module.

We distinguish three general cases that are discussed in the following. The most straightforward case corresponds to a situation in which the erroneous measurement value still belongs to the same high-level context as the corresponding correct context value (cf. figure 5.25). In this case, a detection of the error is not possible since a legal context is observed.

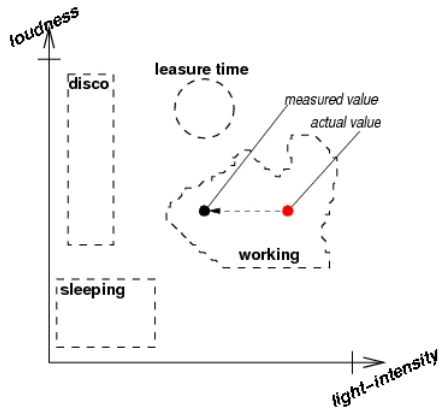


Figure 5.25: Small deviation of error.

Hence an error correction is also not required, since the high-level context after the interpretation step is identical to the one that corresponds to the correct context value. In this case, although the measured value slightly differs from the correct context value, the high-level context is identical. No error correction is therefore required.

In case of the low-level context, although it differs from the low-level context correlated to the correct context value, it corresponds to the identical high-level context. Hence, no difference is evident after the interpretation step has been applied. However, when context prediction is applied on low-level context time series, the erroneous low-level context might cause a context time series that differs from the correct context time series to such degree that an alternative prediction is made for these two time series.

Consequently, although the interpretation step does not actually impose a correction of the erroneous context, the prediction of the high-level context time series might be correct while the prediction on the low-level time series is impaired due to an erroneous input time series. However, this benefit of the high-level prediction case comes at the cost of a higher context abstraction level. As we have discussed in section 5.1.2 the higher context abstraction level leads to a less expressive high-level search space. To summarise, while the higher abstraction level is less susceptible to er-

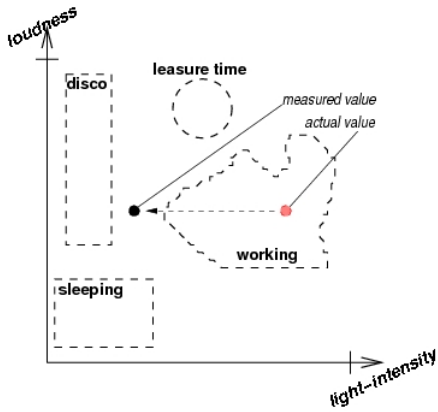


Figure 5.26: Deviation into non-labelled region of the search space.

rors on one hand, the information content is also reduced on the other hand.

The question, which of these two impacts is more serious for the context accuracy, constitutes an interesting open research question.

However, it is also possible that the erroneously measured point falls outside any labelled area in the search space (cf. figure 5.26). In this case a detection of the erroneous measurement is possible and a correction of the error is also achievable. The measurement error is thus so large that the measured value lies outside of the boundaries of the labelled high-level context area. This error can be detected. However, when trying to correct this error, the possibility to incorrectly alter the measured value cannot be avoided. Since the only information available is the measured point and the high-level boundaries, the corresponding correct context value might be any point in the coordinate system. However, a reasonable approach might be to suggest that the correct context value belongs to the high-level region that is closest by. When considering the time as well, it might be even easier to assume the origin of the corresponding correct context value.

Similar to the situation discussed above, for the high-level context case it suffices to assume the right high-level context region, while for the low-

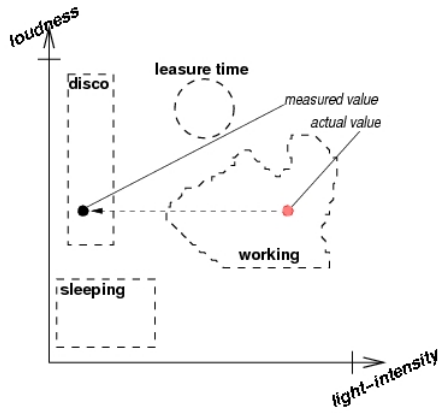


Figure 5.27: Deviation into other labelled region of the search space.

level this might still result in an alternative low-level context time series and in an alternative prediction consequently.

In the third case, the measured point belongs to another labelled set of points than the correct context value does (cf. figure 5.27). In this case a detection of the error is not possible, since the measured point is totally valid given all knowledge available.

Consideration of whole time series When we also consider the time, the error correction abilities increase. For a measurement value that drastically escapes the region in which the recently measured values are located, we might assume that the measured value is erroneous and the corresponding correct context value with high probability is also a member of the recently observed region in the coordinate system. With this technique errors, including those of the third class that fall into neighbouring high-level regions, become distinguishable.

Detection of erroneous context sources Apart from the detection of single measurement errors, the findings discussed in this section might also be utilised to detect erroneous context sources. If, for example, the detected error is typically along one axis, this might indicate a bias in the corre-

sponding context source or sensor. Furthermore, if this error is constantly of similar degree, it might even be possible to correct the error on-line by adding a complementary bias to the measured value of this context source. The same principle can also be applied to multiplicative errors.

Discussion

As we have seen, the improvement of the measured information is actually possible but not restricted to the high-level or low-level context prediction schemes. Both schemes may profit from this correction operation.

Considering the slightly lower correction robustness in case of low-level context prediction, the application scenario will probably be the clincher in favour of the one or the other prediction scheme. If the early detection of trends in the sampled data is crucial, a low-level context prediction scheme is probably most suggestive, while the high-level context prediction scheme is probably more robust against measurement inaccuracies. An in depth study of this topic promises to constitute a most interesting research question. The designer of a context prediction architecture might notice, that the choice of sensors utilised is vital also for the error correction of low-level contexts and that it is hence vital for the overall context prediction accuracy. A choice of sensors that provides a redundant set of context sources may support the error detection process and consequently improve context prediction accuracy.

As a result, labelled high-level context regions that are as far as possible separated from each other and that have a low volume support the error detection capability.

We assume that this detection is even improved if the various context sources attached to a context prediction architecture are chosen in a way that enables a justification or correction of any single output value of a context source based on the output values of all other context sources.

Note that this error detection as described above is not, or only in a restricted manner, possible with an adaptive algorithm. An adaptive algorithm allows for the altering of high-level definitions on-line. If the algorithm designer decides to implement adaptive features into her algorithm, she unfortunately loses (or at minimum weakens) the ability to detect measurement errors by the help of the knowledge available in the context interpretation engine. Further work is required so that the impact may possibly be qualified or reduced.

5.1.3 Simulation results on prediction accuracies

We utilise the context prediction architecture described in chapter 4 to study the impact of the context abstraction level on the context prediction accuracy. The architecture is integrated in Foxtrot, a framework for context-aware computing that was developed by our research group [129]. Foxtrot comprises interfaces of modules for context acquisition, context interpretation and context prediction. We implemented context interpretation and context acquisition modules that enable a tight control of the corresponding error probabilities. The context prediction module utilises our implementation of the alignment search prediction approach (cf. section 4.2.2).

In this section we present simulation results from this simulation that illustrate the properties studied in section 5.1.1 for the analytic case. Further results from the simulations that correspond to other properties are presented in section 5.2.2.

Prediction of GPS-Location data

We study influences of varying levels of context abstraction on the accuracy of high-level and low-level context prediction schemes on a sampled GPS trajectory of a mobile user. We sampled the GPS positions of the user for about three weeks. The sampling hardware consists of a mobile phone and a GPS-receiver. The receiver is connected to the phone via bluetooth. A python script running day and night on the phone was used to obtain the GPS-information from the GPS-receiver.

This GPS data was first stored in the memory of the phone and is afterwards utilised for our study. We took this two-step approach since we extensively analyse high-level and low-level prediction schemes with varying input parameters. A real-time processing would have surcharged the memory and processing capabilities of the phone.

The simulation consists of three consecutive weeks of sampled GPS data from a mobile user. Every 2 minutes a GPS sample is taken. For times when no GPS is available (eg because the user is indoors), we assume that the last available sample best approximates the current position of the user. For the simulation, we utilise the samples on an 8-minutes and 12-minutes scale respectively to reduce sequences of idle periods where no significant movement is observed. In an idle period, it is best to extrapolate the current location of the user into the future. Therefore, if the observed

sequence is dominated by idle periods instead of context changes, methods that utilise this extrapolation receive best prediction results. However, we do not consider this behaviour as learning and prediction, since the really significant context changes are missed although the accuracy of the computed sequence may be high.

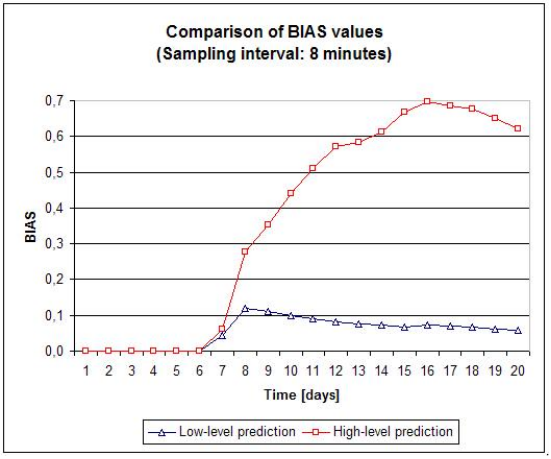
For low-level context prediction we directly utilise the GPS-samples. For the high-level context prediction we define a total of 36 high-level locations as, for instance, 'Home', 'Bakery', 'University', or 'Market'. The high-level locations are specified by a GPS-centre-location and a radius. A default high-level location named 'Outdoors' is applied when no other high-level location matches a low-level GPS sample.

As prediction algorithm, we implement an alignment prediction approach [108] for low-level and high-level context prediction. The context history of the algorithm contains five context elements. It consequently covers a time interval of 40 minutes for the 8 minutes sampling interval and 1 hour for the 12 minutes sampling interval. All parameters of the prediction algorithm are identical for high-level and low-level context prediction.

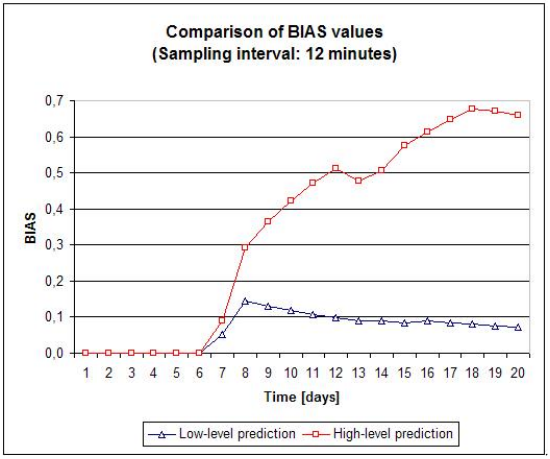
For evaluation of the accuracy we utilise the RMSE and BIAS metrics (cf. section 3.2.4).

Simulation results In figure 5.28 the BIAS values that the low-level and high-level context prediction algorithms scored are depicted. The figures depict the results of the algorithms for a prediction horizon of 15 context elements. This corresponds to a time period of 2 hours for the 8 minutes sampling interval and a prediction horizon of 3 hours for the 12 minutes sampling interval. For both prediction schemes, the errors in the first six days of the simulation are minor. This is because during this time the user had a work week in which the way she took to work at all days did not differ much from each other. When the first weekend started, the user behaviour changed and new time series, that had not been observed by the algorithm so far, occurred. Therefore, the BIAS values increase drastically for high-level and low-level context prediction. At the time of the second weekend, we again observe an increase in the BIAS values, although it is by far less harsh than the first one.

Generally, the low-level context prediction scheme performs better than the high-level context prediction scheme, although the input data is identical for the high-level and the low-level context prediction process.



(a) BIAS values at a sampling interval of 8 minutes.



(b) BIAS values at a sampling interval of 12 minutes.

Figure 5.28: Context prediction accuracies. The BIAS-values of low-level and high-level context prediction schemes differ from one another for various sampling intervals.

The most significant difference between the different sampling intervals is the performance of the low-level context prediction algorithm which is slightly decreased at a sampling interval of 12 minutes.

Comparing both prediction schemes, we observe that the low-level context prediction scheme outperforms the high-level context prediction approximately by factor 10 (cf. figure 5.29) regardless of the sampling interval. However, this is only true in the second half of the simulation. At the beginning of the simulation in the first week, when only little data has been observed, the high-level context prediction scheme performs better. As a reason for this we account the complexity of the context time series. Due to the higher context abstraction level of the high-level context history, the patterns observed are more general and of a simpler structure. At times where only few, easily distinguishable patterns exist, this constitutes a benefit. However, with the introduction of further context time series that are harder to distinguish between, the higher context abstraction level becomes a hindrance. As we observe in figure 5.28, the prediction errors at the beginning of the simulation are only minor for both prediction schemes.

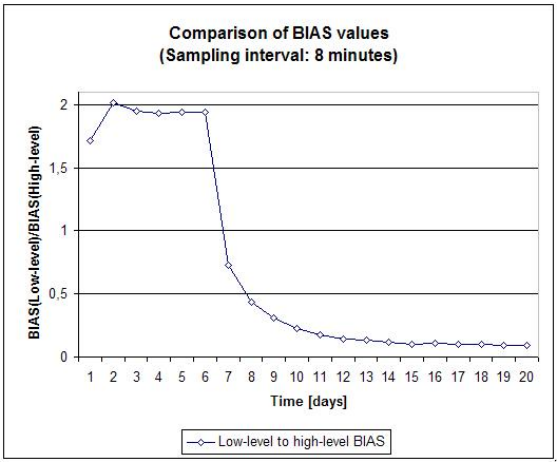
For a sampling interval of 12 minutes, the high-level context prediction scheme performs slightly better in direct comparison to the low-level context prediction scheme. However, the general properties described above remain unchanged.

When considering the RMSE metric, the results are similar (cf. figure 5.30). The only difference we observe is that the factor by which the high-level prediction is outperformed by the low-level context prediction scheme is slightly smaller (cf. figure 5.31). These results are not surprising since the analytical findings in section 5.1.1 suggested these trends.

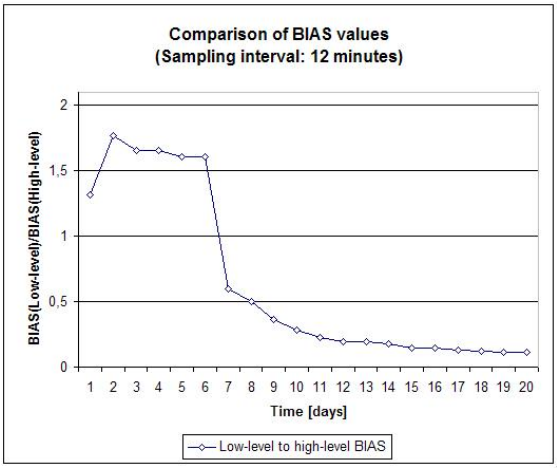
When the context prediction horizon is modified, these general trends stay evident (cf. figure 5.32). We can, however, observe another property that we also observed in the analytical discussion. With increasing context prediction horizon, the advantage of the low-level prediction algorithm over the high-level context prediction algorithm increases.

Finally, we conduct simulation runs in which we modify the context history length. For these simulations we choose a sampling interval of 20 minutes and a context history length of 10 context elements (200 minutes), 15 context elements (300 minutes) and 30 context elements (600 minutes) respectively.

With an increasing context history length, the performance gain of the

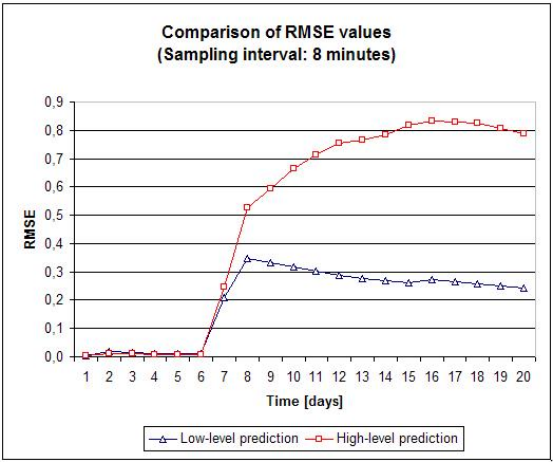


(a) BIAS values at a sampling interval of 8 minutes.

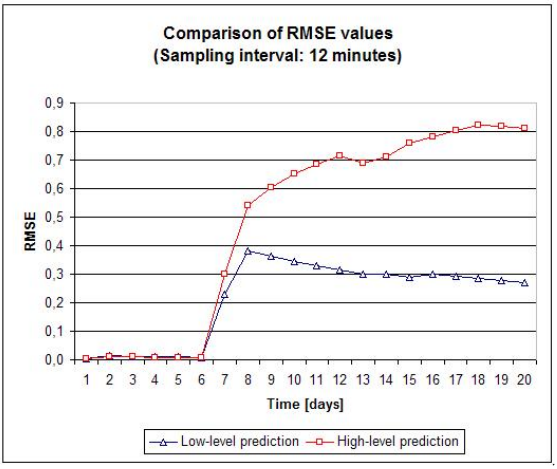


(b) BIAS values at a sampling interval of 12 minutes.

Figure 5.29: Comparison of the context prediction schemes. The figures show the fraction of low-level BIAS divided by high-level BIAS for various sampling intervals.

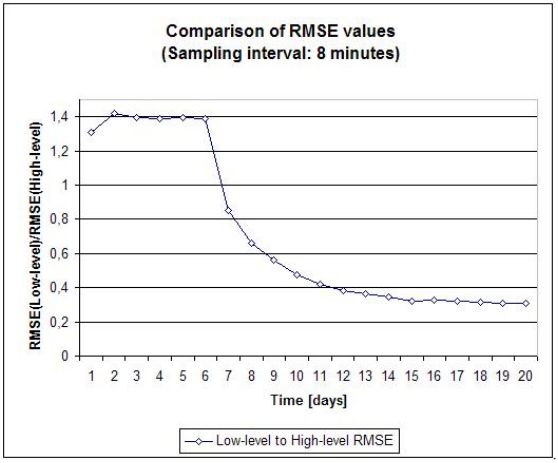


(a) RMSE values at a sampling interval of 8 minutes.

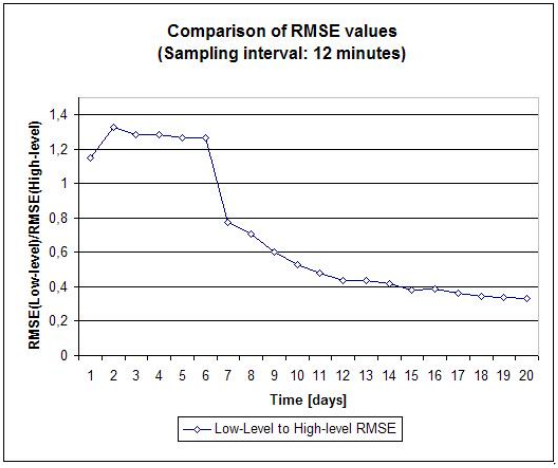


(b) RMSE values at a sampling interval of 12 minutes.

Figure 5.30: Context prediction accuracies. The RMSE-values of low-level and high-level context prediction schemes greatly differ from one another for various sampling intervals.



(a) RMSE values at a sampling interval of 8 minutes.



(b) RMSE values at a sampling interval of 12 minutes.

Figure 5.31: Comparison of the context prediction schemes. The figures show the fraction of low-level RMSE divided by high-level RMSE for various sampling intervals.

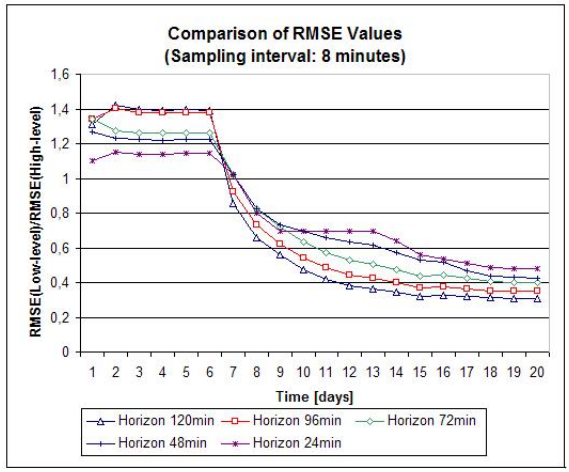
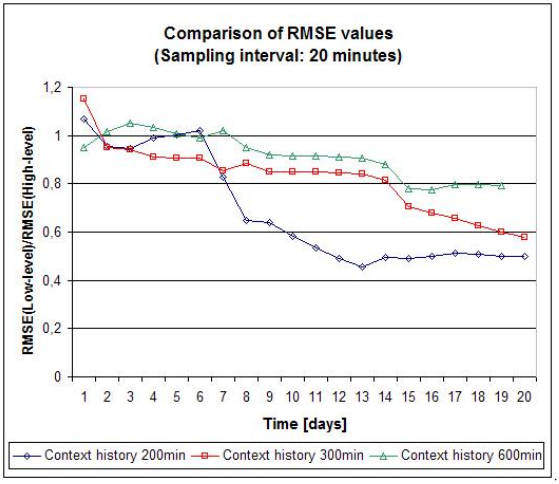


Figure 5.32: RMSE Values for varying prediction horizons.

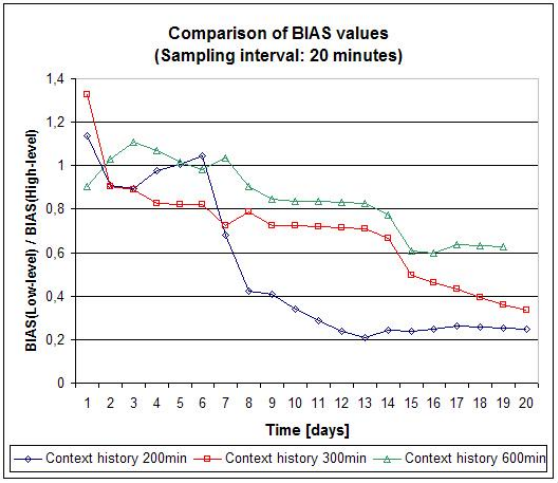
low-level context prediction scheme over the performance of the high-level context prediction scheme decreases. Although this property was not evident from the analytic consideration in section 5.1.1, we have argued in section 5.1.2 that the low-level context prediction scheme is better capable to cope with a shorter context history since the information content of the low-level search space is higher than the information content of the high-level context prediction search space. A simple method to increase the expressiveness of a time series is to increase its length. With longer observed context time series the impact of the search space dimension on the information content of the time series decreases. The results for the RMSE and BIAS metrics are depicted in figure 5.33.

In summary, we have observed that the low-level context prediction scheme is advantageous when compared to the high-level context prediction scheme on this location data set. Furthermore, we could observe that the impact of an increasing prediction horizon is more serious for high-level than for low-level context prediction, as it has been suggested by the analytical discussion in section 5.1.1.

Finally, an effect that could not be observed by the analytical discussion



(a) Comparison with regard to the RMSE metric.



(b) Comparison with regard to the BIAS metric.

Figure 5.33: Comparison of low-level and high-level results.

since it does not originate from context processing errors, but from different search spaces for low-level and high-level context prediction schemes, is the impact of the context history size. For an increasing context history size, the accuracy gap between the accuracies of the high-level and low-level context prediction schemes is narrowed. While low-level context prediction schemes cope better with short context histories, this advantage diminishes with an increasing context history size.

Influence of the context interpretation error

A further property evident from the analytic simulation that could not be observed in the GPS-simulation in the last chapter is the influence of the interpretation error on the context prediction accuracy. For interpretation errors that exceed the context prediction error, we expect the low-level context prediction scheme to outperform the high-level context prediction scheme. In the realistic GPS simulation, the impact of the interpretation error is smeared together with the impacts of the acquisition and prediction errors. This is an implicit drawback of any realistic simulation. In realistic scenarios of considerable size it is hardly possible to isolate or especially to identify the various impacts on the simulation result. We therefore isolate the context interpretation error from other errors and further exclude side effects that might influence the simulation.

In the GPS scenario described in section 5.1.3, all data utilised has been sampled under real-life conditions. Although the results therefore demonstrate the applicability of context prediction and the benefits of low-level context prediction also in realistic scenarios, various side effects that are not considered in the simulation setting might influence the results.

These effects are known as concept drift and can be summarised as hidden changes in contexts as described in [130]. Concept drift is often associated with changes in a context that is hidden from the context-aware system. Examples for concept drift in a financial setting are the current interest rate or the market mood. In the GPS-example above, the change of habits or new friends or project partners might constitute a concept drift.

In order to obtain a more complete understanding of the influence of the context interpretation error, we exclude hidden changes in context due to concept drift from the simulation by conducting a simulation of the prediction algorithms on synthetic data. In this data set, the low-level con-

text patterns are manually created and contain special properties that are suitable for the illustration of a desired effect. We further exclude the context acquisition step to focus on the impact of the context interpretation step only. This is done by the consideration of already acquired low-level context patterns instead of raw data. To exclude further side effects, the learning process of the context prediction algorithm is disabled.

We again utilise the Foxtrot implementation but abstract from the context acquisition module. A new context interpretation module has been implemented for this simulation.

For the context interpretation module, we provide an exact mapping between the low-level and high-level contexts. For each low-level context we represent a unique high-level context. The error probability of this module is configurable so that it enables us to control the context interpretation error applied.

The context prediction task is accomplished by the alignment prediction implementation. All prediction parameters of this algorithm are chosen identically for high-level and low-level context prediction schemes. In case of low-level context prediction, the context interpretation step is applied on the predicted low-level context time series. For high-level context prediction, the low-level time series are first transformed to high-level context time series, while the prediction is applied thereafter. The context interpretation error is varied in different simulation runs from 0.1 to 0.5. We decided for a uniform distribution of errors. In this simulation, we describe the accuracy by the fraction of accurately predicted contexts to the number of predicted contexts. This measure of accuracy is directly linked to the error probability in the prediction context time series. With an interpretation error of 0.0, the alignment prediction method has a prediction accuracy that is close to zero in this small scenario.

We utilise four distinct low-level context patterns with 41 elements each. The patterns are one-dimensional context time series of real valued context elements. Two patterns are increasing and two are decreasing with partly redundant contexts and varying slope. Patterns 1 and 2 contain increasing values from 0 to 20 and from 0 to 40 respectively while for patterns 3 and 4 the values decrease in regular intervals from 40 to 0 and from 20 to 0. In the course of the experiment we repeatedly choose one of these patterns with a uniform distribution and feed it into Foxtrot.

Interpretation error	correct predictions	accuracy
0.1 (high-level)	1405	0.964
(low-level)	1310	0.898
0.2 (high-level)	850	0.583
(low-level)	1160	0.796
0.3 (high-level)	649	0.445
(low-level)	1024	0.702
0.4 (high-level)	482	0.331
(low-level)	882	0.605
0.5 (high-level)	25	0.017
(low-level)	693	0.475

Table 5.1: Context prediction accuracy.

Simulation results The results of this simulation are illustrated in Table 5.1. We observe that with a context interpretation error of 0.1, the high-level prediction accuracy is higher than the low-level prediction accuracy. However, with a context interpretation error of 0.2 or higher, the low-level prediction method achieves the better accuracy. These results again support our analytical studies from which we expected the low-level context prediction scheme to outperform the high-level context prediction scheme for interpretation errors that exceed the context prediction error.

The corresponding ratios of the low-level to high-level accuracies are depicted in Table 5.2. Most significantly, these values show a drastic decrease in the high-level context prediction accuracy. While it might be still feasible for some applications to utilise the low-level context prediction scheme with low context interpretation accuracies, the prediction accuracy of the high-level context prediction scheme diminishes at such a fast pace that it quickly becomes infeasible for arbitrary applications.

Time series dimension and context history size

A further property that we were not able to show in the GPS-simulation in section 5.1.3 is the influence of the time series dimension of the input

Interpretation error	Ratio of prediction accuracies
0.1	1.073
0.2	0.732
0.3	0.634
0.4	0.547
0.5	0.036

Table 5.2: High-level to low-level context prediction accuracy.

time series. The reason for this is the lack of different hardware sensors that were available in our case. We therefore choose again a synthetic data set for simulation purposes.

In the current section, we study the influence of a varying number of raw data values utilised and also vary the size of the context history. In this synthetic simulation, the same modules as in the synthetic simulation study of the interpretation error above are utilised. Additionally, an acquisition module that allows a tight control of the acquisition error is utilised. This implementation is basically identical to the implementation of the interpretation module. Furthermore, we increase the time series dimension. In the simulations a maximum of 10 dimensions are applied. We utilised 12 different time series of raw data values for each dimension, resulting in 120 distinct time series overall. These time series contain real valued context elements and are not congruent. The acquisition and interpretation error probabilities are set to $P_{acq} = 0.98$ and $P_{int} = 0.94$ respectively.

For the interpretation error we assume a uniform distribution of possible errors, while we apply a Gaussian distribution for the acquisition error. The Gaussian distribution models the property that small errors are more reasonable than substantial errors in the acquisition module. Table 5.3 summarises the configuration of the simulation environment.

The prediction modules for high-level and low-level context prediction were trained with these time series in advance. In each simulation run we choose 12 context time series following a uniformly random distribution one after another out of the pool of time series and subsequently fed them into the architecture. For low-level context prediction the time series are

Parameter	Value
Number of time series	12
Length of the time series (elements)	41
Minimum value for any Raw data	0.0
Maximum value for any raw data	100.0
Time series dimension (DIM)	$1, 5, 10$
Context history size (LEN)	$5, 10, 15, 20$
Simulation runs per (DIM,LEN) pair	12
Acquisition accuracy (Gaussian)	$P_{acq} = 0.98$
Interpretation accuracy (Uniform)	$P_{int} = 0.94$

Table 5.3: Configuration of the simulation environment.

Context history size	5	10	15	20
Dimension 1	0.93	0.62	0.58	0.25
Dimension 5	0.96	0.90	0.69	0.64
Dimension 10	0.96	0.94	0.91	0.90

Table 5.4: Ratio of error probability (P_{ll}/P_{hl}).

first processed by the context acquisition modules and are then input into the prediction component. The output time series of the prediction component are further forwarded to the context interpretation module. For high-level context prediction the time series are, after being processed by the acquisition component first handled by the context interpretation component and are only then processed by the context prediction component.

Simulation results The simulation results are summarised in table 5.4. In the table we depict the fraction of the results obtained by the context prediction based on low-level context elements to the results obtained by high-level context prediction scheme. The figures in the tables give an estimation on the degree of predominance one prediction scheme has over the other one.

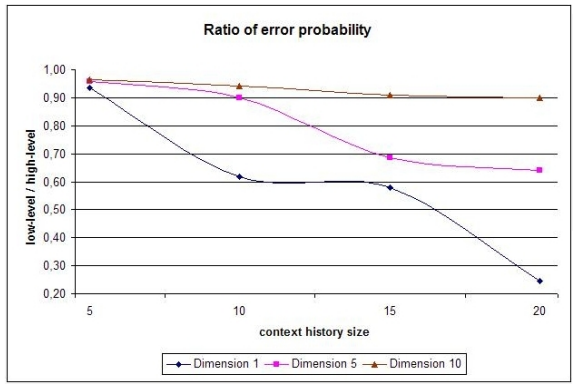
In the table, the probability of error is compared for high-level and low-level context prediction schemes. The impact of an increasing context

history size is evident from these simulation results. We observe that the general trends we expected due to our results from the analytic study in section 5.1.1 are confirmed by the simulation. With increasing time series dimension, the predominance of the low-level context prediction scheme above the high-level context prediction scheme diminishes. Furthermore, with increasing context history size the predominance of the low-level context prediction scheme above the high-level context prediction scheme increases. At first glance, this property contradicts our observation of the influence of the context history in section 5.1.3. In the simulation on GPS location data the low-level context prediction scheme showed an improved performance with a shorter context history. The reason identified for this property was a different size of the high-level and low-level search spaces. In section 5.1.3, the context interpretation altered the search space dimension of the prediction algorithm while in the current consideration this effect is nullified by the special property of the interpretation module. For this reason, the influence of the search space dimension is not evident in this simulation. As stated above, the interpretation module non-ambiguously maps every point in the low-level search space to an unique point in the high-level search space. Consequently, the sizes of both search spaces are identical.

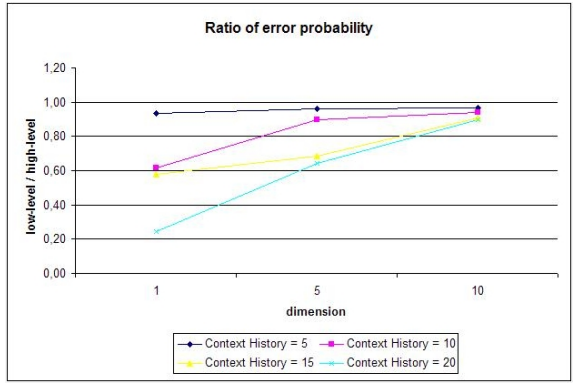
What more seriously impacts the prediction accuracy in this case is the number of erroneous contexts in the input time series. This value is higher for high-level context prediction schemes and increases with increasing context history length. With more errors in the input time series the context prediction accuracy consequently decreases.

For increasing size of the context history, the low-level context prediction scheme is increasingly superior to the context prediction based on high-level context elements. This trend is illustrated in figure 5.34(a). While the error probability for both prediction methods is nearly equal with a context history of 5 elements, the context prediction based on low-level context elements is clearly advantaged when the size of the context history increases.

Another trend visible in table 5.4 is that the dominance of the context prediction based on low-level context elements diminishes with increasing dimension of the context history. This again confirms the corresponding result obtained in the analytic consideration in section 5.1.1. With increasing context time series dimension the high-level and low-level context prediction schemes dominate an equal share of the total probability space



(a) Number of context sources to the context history size.



(b) Context history size to the number of context sources.

Figure 5.34: Ratio of low-level to high-level context prediction accuracy.

that describes the prediction accuracy. This property can be observed from figure 5.34(b).

5.1.4 Summary

We have observed that the prediction accuracies of the high-level and low-level context prediction schemes differ from one another. The two main reasons that have been identified to account for this property are the higher probability of error for high-level context histories as well as the different sizes of the search spaces utilised for high-level and low-level context prediction schemes. Parameters that constitute the main difference between high-level and low-level context prediction schemes are the dimension of the input time series, the size of the context history as well as the size of the prediction search space. Overall, we could observe a strong advantage of the low-level context prediction scheme in all scenarios and approaches discussed. We therefore recommend that the application designer also considers various context prediction schemes in order to improve in accuracy.

All results have been obtained in an analytic consideration first and have been confirmed in simulation on real and synthetic data.

In a simulation conducted on the GPS-data of a mobile user, we observed that the prediction of the low-level context prediction scheme had the lower deviation from the actually observed contexts than the high-level context prediction scheme.

In synthetic simulations we have been able to observe further effects of the environmental parameters discussed in the analytic consideration. These are the size of the context history, as well as the time series dimension and the interpretation error.

For the length of the context history we could observe two effects. Due to a smaller size of the search space, high-level context prediction is disadvantaged for small context history sizes. Furthermore, because of the increased error probability of the input time series, after the context interpretation step has been applied, the high-level context prediction scheme is again disadvantaged.

5.2 Learning and long-term accuracy

Provided that a context source is in good order, the low-level context elements reflect the actual situation with respect to the acquisition inac-

curacy. By interpreting low-level context elements to high-level context elements, we cannot exclude an erroneous context interpretation.

A prediction based on high-level context elements might therefore be based on information that is more likely to be erroneous than low-level context elements. This does not only affect the instantaneous prediction accuracy, as discussed in section 5.1, but may also affect the long-term prediction accuracy if the context prediction method implements a constant learning procedure in order to adapt to a changing context evolution process. The learning procedure is based on the observed context elements which is more likely to be erroneous in case of high-level context elements.

In section 3.2 we defined context prediction as the task of learning a function that describes the context evolution in time. The ability to learn frequent patterns in human behaviour is therefore tightly linked to context prediction. However, regardless of the learning method utilised, the learning accuracy might suffer from basic properties of the input data. For high-level context prediction, the input data is a time series of high-level context elements while for low-level context prediction it is a time series of low-level context elements.

The reliability of high-level context time series and low-level context time series differs from one another. The reason for this is the context interpretation step that transforms low-level context elements to high-level context elements. On the one hand, information contained in the input data might be lost in the interpretation step, on the other hand, this interpretation step might induce further errors in the context elements as we have discussed in section 5.1.1 and in section 5.1.2.

In the following sections we discuss issues of the data abstraction level on context credibility and the impacts on the learning process of the context prediction scheme.

In section 5.2.1 the situation is first considered analytically before we introduce simulation results in section 5.2.2 that confirm the analytic findings.

5.2.1 Analytical discussion

The context prediction procedure for low-level and high-level context prediction respectively is depicted in figure 5.35. For low-level context prediction, the sequence of operations includes the context acquisition, followed by the context prediction and context interpretation. For high-level con-

text prediction on the other hand, context interpretation is applied in advance of context prediction. Since context prediction requires the learning of context elements, the error probability of the input data influences the learning capability.

The sources of error that impact the accuracy of the high-level and low-level input time series are the acquisition and the interpretation procedures. Error probabilities relevant in our case are

P_{acq} The probability that no error occurs in the context acquisition step.

P_{int} The probability that no error occurs in the context interpretation step.

Let $i, k, m, o \in \mathbb{N}^{\setminus 0}$. Assume that the context prediction method bases its calculation on k context elements, regardless of the prediction scheme (high-level or low-level) and that each element in the low-level time series is composed of m low-level contexts, while each element in the high-level time series is composed of o high-level contexts. We define an error in one of the context processing steps as an incorrect interpretation or aggregation of context elements. In our discussion, assumptions taken for this process are the same as in section 5.1.1.

We obtain the error probabilities for low-level and high-level context time series analogous to the derivation in section 5.1.1. The low-level context time series is erroneous with a probability of

$$P_{ll}^{TS} = 1 - P_{acq}^{km} \tag{5.14}$$

while the high-level time series on the other hand has an error probability of

$$\begin{aligned} P_{hl}^{TS} &= 1 - \left(P_{acq}^{m} \cdot P_{int}^{o} + (1 - P_{acq}^{m})(1 - P_{int}^{o}) \frac{1}{v_h^o - 1} \right)^k \\ &= 1 - P_{acq}^{km} \cdot \left(P_{int}^{o} + \frac{(1 - P_{acq}^{m})(1 - P_{int}^{o})}{P_{acq}^{m} \cdot (v_h^o - 1)} \right). \end{aligned} \tag{5.15}$$

Since $P_{ll}^{TS} < P_{hl}^{TS}$, the high-level context time series is more likely to be error prone.

When context prediction is based on high-level contexts, the learning procedure operates on a time series that is more likely to contain errors than in the low-level prediction case. We therefore expect the learning in

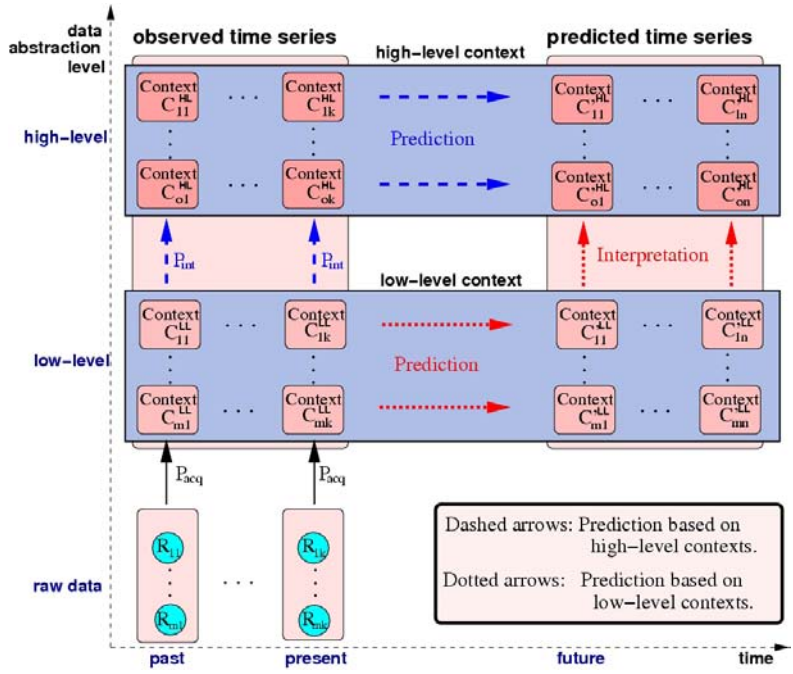


Figure 5.35: Prediction based on high-level or on low-level context elements respectively.

case of low-level context prediction to perform better than for high-level context prediction.

In the following sections we investigate the impact of errors in the input time series on the learning accuracy.

Impact of interpretation errors on the learning task

In this section we analyse the impact of errors in the input sequence on learning approaches that are optimal in the sense that they take with every action the most reasonable decision in order to obtain an optimal learning accuracy. Algorithms that follow another notion of optimality or that make errors considering the optimisation aim and the information available are not considered.

The task of the learning method for context prediction tasks is to find the mapping between the currently observed time series to the time series that will be observed next (cf. section 3.2.4). We describe the state of the learning algorithm by the mapping between observed and predicted states, the algorithm currently follows. In the course of the computation, the algorithm might change its state, which means that it alters the mapping between observed and predicted time series. The maximum number of states an algorithms can possibly take is consequently given by the number of valid mappings.

With every observed context, the state of the algorithm might change. We illustrate the deterministic learning process as a graph $G = (V, E)$ where V is the set of states and E is the set of edges describing the transition between states (cf. figure 5.36). When the observed and predicted time series have a finite length, this graph is finite since both the number of states and the number of transitions are finite.

We rate the states according to their accuracy for the prediction process. One state is better than another state if the prediction accuracy is higher. Observe that it is only possible to state this rating with the knowledge of the correct context evolution process. However, although we might not be able to provide a rating for the states in real time, due to the lack of this knowledge, such a rating is existent at all times. The higher the ratings of the states an algorithm typically operates on, the better.

We consider now the impact of errors in the input sequence on the transition of the learning algorithm on these states. In other words, with this knowledge we are able to estimate the influence of the interpretation

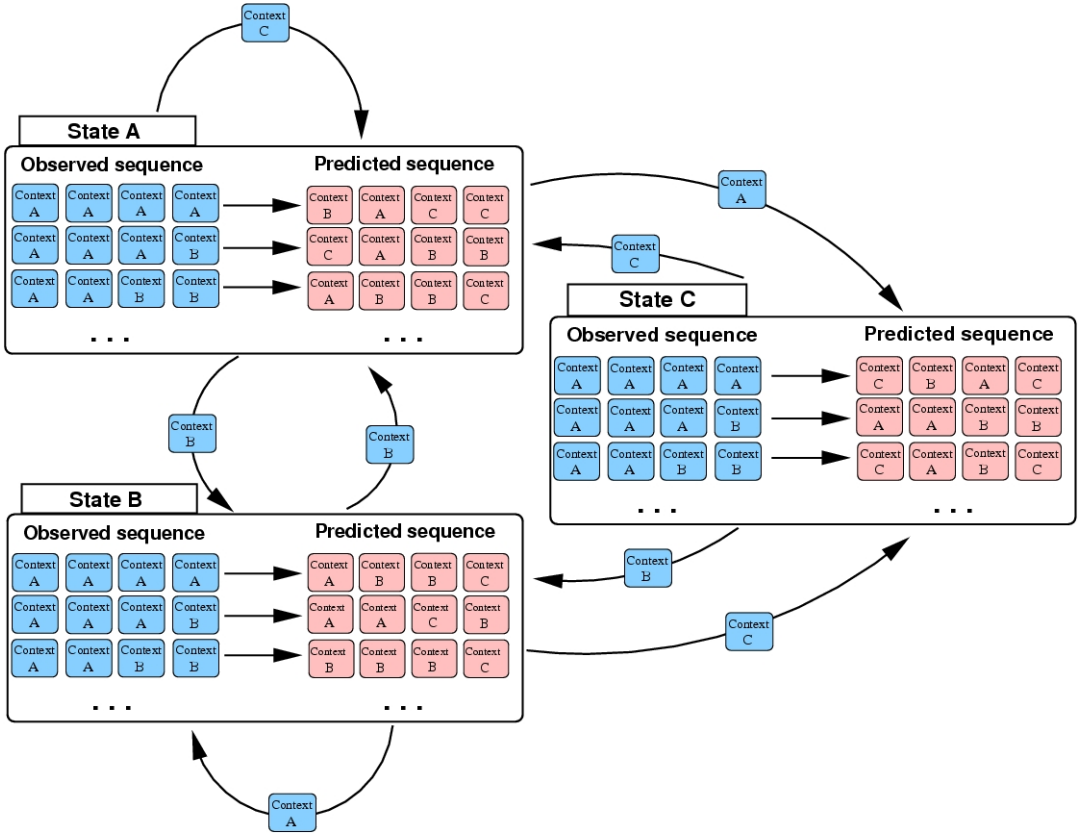


Figure 5.36: Transitions between various states

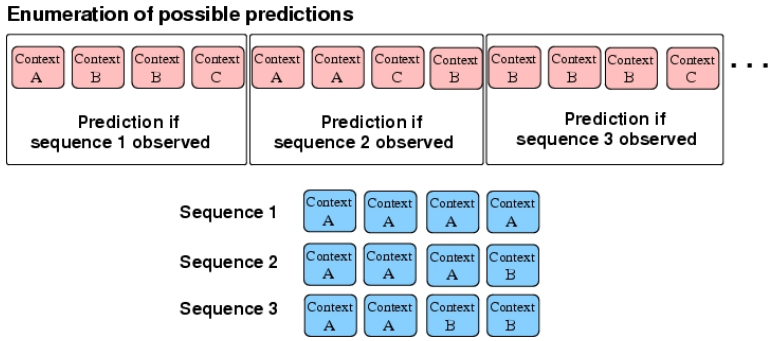


Figure 5.37: Enumeration of possible predictions as encoding.

errors on the ratings of the states in the state transition sequence.

Clearly, suboptimal algorithms or insufficient knowledge of the context evolution process might also affect the state transition sequence. In order to exclude these side effects in our consideration we assume that all information necessary to describe the context evolution process is available to the learning algorithm. Remember that we consider an optimal learning algorithm. This algorithm is optimal in the sense that, if all information necessary to describe the context evolution process is available, it will always evolve to the highest rated state.

In other words, we assume that all information necessary to evolve to the highest rated state is encoded in the input sequence. An example for a possible, although unlikely encoding is depicted in figure 5.37. This encoding non-redundantly enumerates all fixed length predictions for every possible input sequence of fixed length in a predefined order.

When the input sequence is erroneous, errors might be induced into the encoded information which in turn might result in the prediction algorithm to evolve to another state than the highest rated one. Generally, the information describing the context evolution process might contain redundant information so that not every error is serious. Several errors in these partly redundant context time series have to occur in order to result in a severely altered context sequence that can not be retrieved correctly.

Assume that a single context c_i in the input sequence is guarded by s_i redundant contexts. Let P_{err} be the probability that an arbitrary context

in the input sequence is erroneous. With probability $P_{err}^{s_i}$ this error cannot be corrected, since all corresponding redundant contexts in the input sequence are equally erroneous.

Since the probability of error for low-level contexts is lower than for high-level contexts, the high-level learning method is more likely to operate in lower rated states that generate more prediction errors.

5.2.2 Simulation results on impacts on the learning accuracy

In the following sections we present results related to the learning accuracy of high-level and low-level context prediction schemes that have been obtained in simulations. These simulations are conducted on real and symbolic data sets respectively.

Location prediction

We study influences of varying levels of context abstraction on the learning capabilities of high-level and low-level context prediction schemes on a sampled GPS trajectory of a mobile user. The simulation scenario is identical to the scenario described in section 5.1.3.

We conduct two simulation sequences. In the first simulation, no prior learning is allowed. In a second implementation, we utilise the first half of the data for learning purposes. For the learning approach we decided to identify typical context patterns to be stored in the rule base (cf. section 4.2.2). Patterns in the observed context sequence that significantly differ from all recently observed typical context patterns in the rule base are added to the rule base as additional typical context patterns.

A weighting or removal of more or less important patterns is not considered since the simulation time only lasted three weeks. A significant change in habits is not expected for the user in this short time period.

In the simulations, the data has been sampled at varying intervals and with various prediction horizons. We depict simulation results for sampling intervals of 8 minutes and 12 minutes respectively. The prediction horizon covers periods of 48 minutes, 92 minutes and 120 minutes. As we have discussed above we expect the accuracy improvement due to the learning approach to be less evident for high-level context prediction schemes. Prediction accuracies are measured by means of the BIAS and RMSE metrics (cf. section 3.2.4).

	Low-level		High-level	
	RMSE	BIAS	RMSE	BIAS
No learning	0.2693	0.0725	0.8122	0.6596
learning	0.2347	0.0551	1.3375	1.7888

Table 5.5: Prediction accuracies with a prediction horizon of 120 minutes and a sampling interval of 12 minutes.

	Low-level		High-level	
	RMSE	BIAS	RMSE	BIAS
No learning	0.2438	0.0595	0.7881	0.6211
learning	0.1996	0.0398	1.2558	1.5771

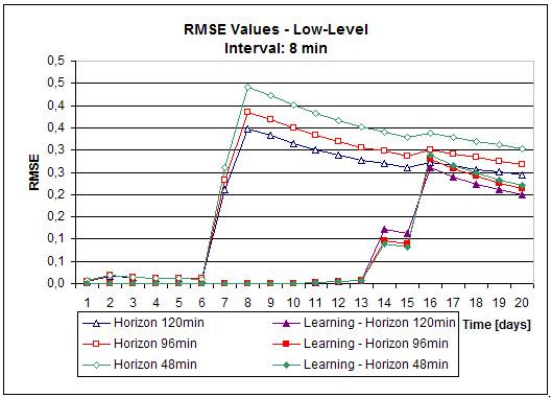
Table 5.6: Prediction accuracies with a prediction horizon of 120 minutes and a sampling interval of 8 minutes.

Simulation results Table 5.5 and table 5.6 depict the prediction accuracies for high-level and low-level context prediction schemes for varying sampling intervals.

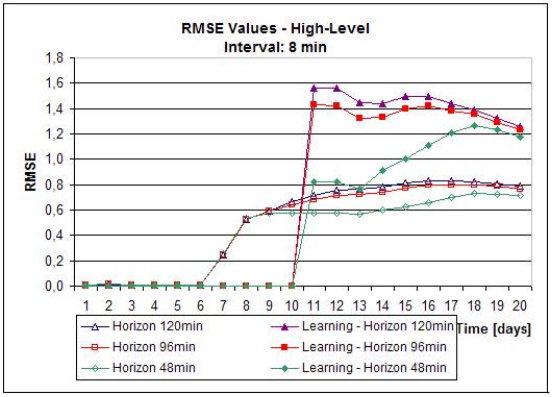
For low-level context prediction, a learning prior to the simulation improves the simulation result with both sampling intervals. For high-level context prediction however, we observe that the overall result is worse than for the low-level case. Actually, the learning approach even worsens the prediction result. We observe this property also in figure 5.38 for a sampling interval of 8 minutes and in figure 5.39 for a sampling interval of 12 minutes.

In these figures, the curves labelled 'Learning - Horizon χmin' depict the results when learning is utilised for a prediction horizon of χ minutes. The curves labelled 'Horizon χmin' depict corresponding results without learning. Since the learning algorithm utilised the first half of the data for learning purposes, results are only available for the second half of the simulation.

In the figures, the RMSE values are depicted for low-level and high-level context prediction schemes. As described above, the low-level context prediction scheme benefits from the learning process while the high-level context prediction scheme even suffers from it. However, regarding the

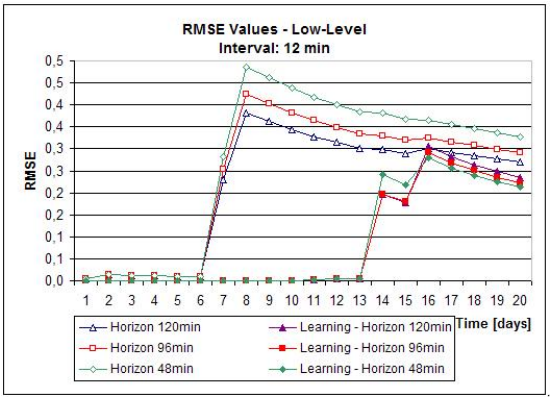


(a) RMSE values for the low-level context prediction scheme.

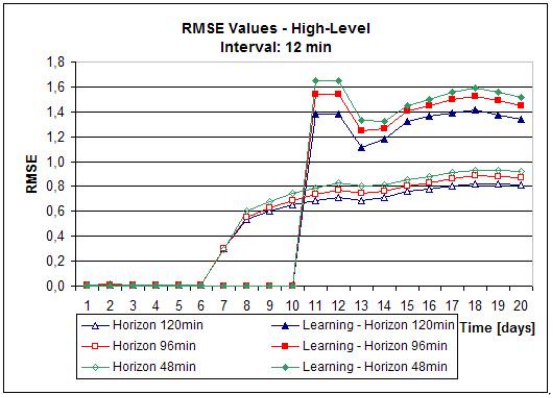


(b) RMSE values for the high-level context prediction scheme.

Figure 5.38: Comparison of low-level and high-level context prediction schemes at a sampling interval of 8 minutes.



(a) RMSE values for the low-level context prediction scheme.



(b) RMSE values for the high-level context prediction scheme.

Figure 5.39: Comparison of low-level and high-level context prediction schemes at a sampling interval of 12 minutes.

sampling interval, the high-level context prediction algorithm obtains better results for a higher sampling interval of 12 minutes while the low-level context prediction scheme performs slightly worse for the higher sampling interval. For various prediction horizons, the RMSE values are for both prediction schemes decreasing with increasing learning time as expected.

In figure 5.40 these results are compared directly to each other. The figure shows the fraction of low-level RMSE values to high-level RMSE values for both sampling intervals.

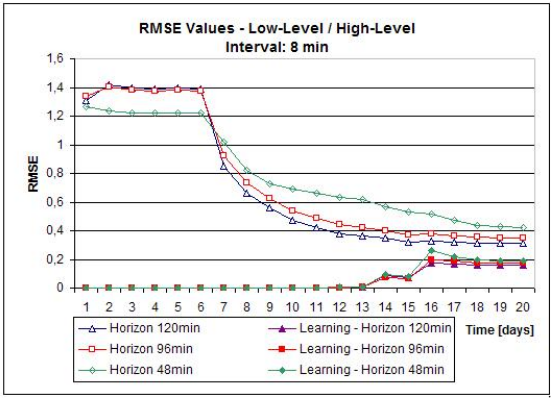
From the time-evolution we observe that the accuracy of the high-level context prediction scheme is above the low-level context prediction scheme for the first four days. This is due to the lower complexity of the high-level contexts. However, in the pace of the experiment, the low-level context prediction outperforms the high-level context prediction due to the increased search space size and consequently the higher information content of the low-level contexts. Similar results can also be observed for the BIAS metric (cf. figure 5.41 and figure 5.42).

These results could be expected since the analytic consideration in section 5.2.1 also suggested a better performance of the low-level context prediction scheme since the error probability of the low-level context time series is lower. Furthermore, we analyse the amount of memory required by the learning approach for low-level and high-level context prediction schemes. For every typical context pattern observed, the learning algorithm stores a time series in the rule base. As a measure for memory consumption, we count the number of time series in the rule base and the number of context values in all rule base time series (cf. figure 5.43).

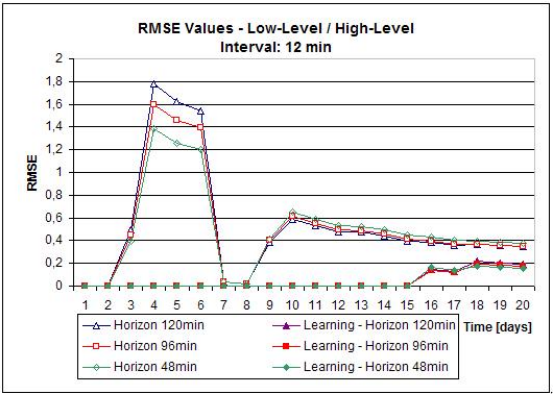
We observe that the ratio of time series and context elements for low-level to high-level prediction schemes increases over time. After a simulation time of 20 days the low-level context prediction scheme stores about 1.5 times the amount of context values as the high-level context prediction scheme. The number of context values per time series is more or less constant but approximately 1.1 times higher for low-level context prediction.

Note that these values also influence the processing load, since the algorithm processes every single time series in the rule base in order to find alignments most similar to the observed time series.

We therefore conclude that the low-level prediction scheme has higher memory and processing requirements, although the simulation parameters are equal. As a reason for the higher count of context patterns we account the differing context abstraction levels of high-level and low-level context

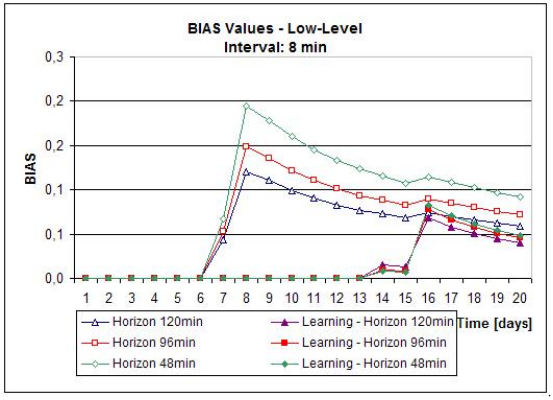


(a) 8 minutes sampling interval.

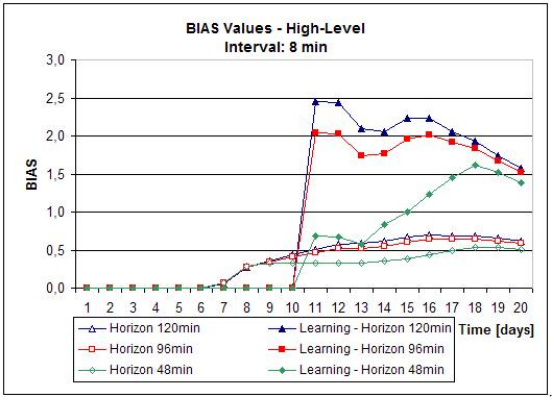


(b) 12 minutes sampling interval.

Figure 5.40: Fraction of the low-level to high-level RMSE values.

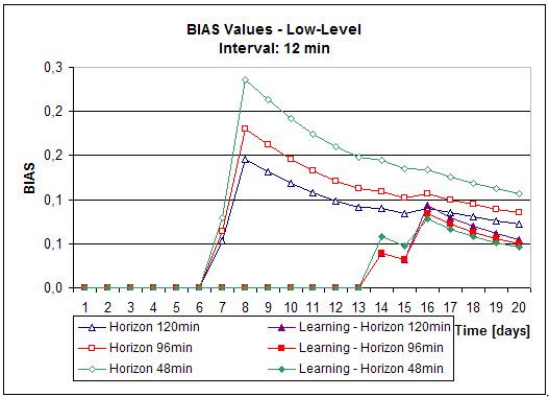


(a) Low-level results for the BIAS metric.

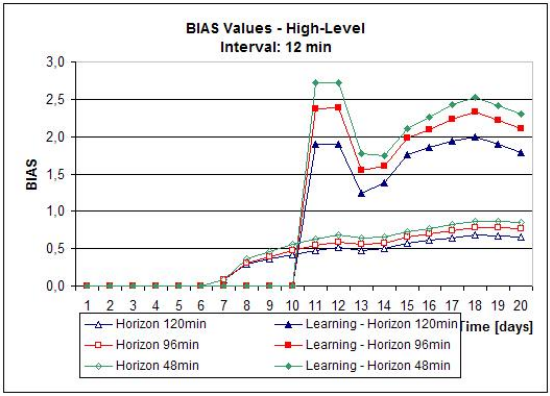


(b) High-level results for the BIAS metric.

Figure 5.41: Comparison of context prediction schemes for an 8 minutes sampling interval.



(a) Low-level results for the BIAS metric.



(b) High-level results for the BIAS metric.

Figure 5.42: Comparison of context prediction schemes for an 12 minutes sampling interval.

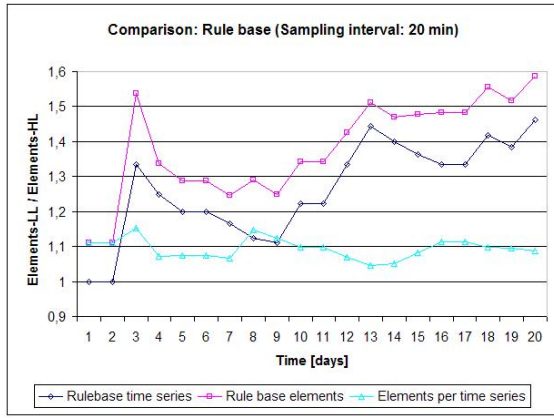
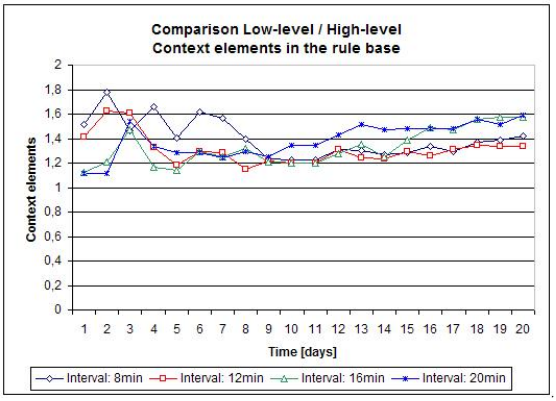


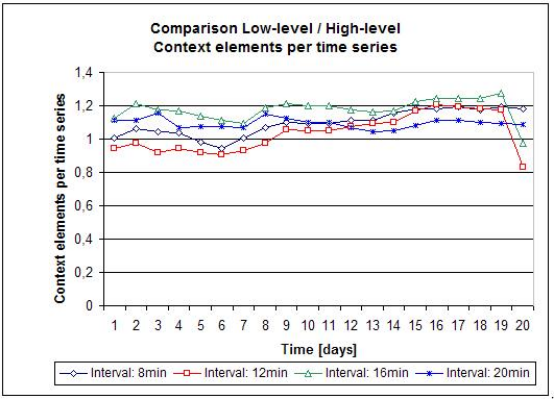
Figure 5.43: Contents of the rule base for low-level and high-level context prediction schemes for various prediction times.

time series. Some differing low-level time series might be represented by very similar high-level context time series. Consequently, in the high-level case, only one representative typical context pattern is stored to the rule base while in case of low-level context prediction it might be several representative context patterns. Figure 5.44 shows for low-level and high-level context prediction schemes the fraction of the number of context elements stored in the rule base and the number of context elements per time series in the rule base for various sampling intervals. Obviously, the memory consumption of the low-level context prediction scheme is higher for all sampling intervals. The same is true for the number of time series stored in the rule base and for the context elements stored for every one context element that had been observed (cf. figure 5.45).

To summarise, we could observe that the learning approach for high-level context prediction schemes has proved to be even counterproductive since it decreases the prediction accuracy. For a low-level context prediction scheme we have observed, however, that the learning applied to the prediction process improved the prediction accuracy. These results confirm our findings in the analytical discussion in section 5.2.1. In spite of that, regarding the memory consumption and processing load, we ob-

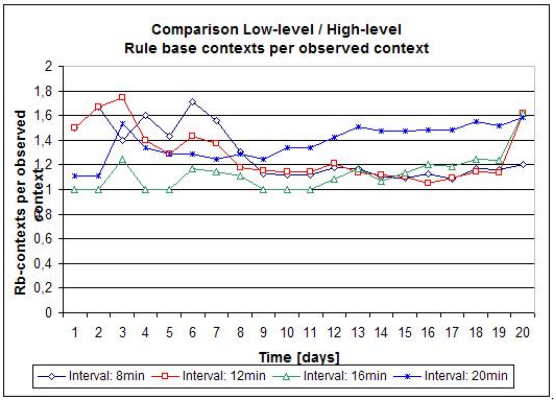


(a) Context elements in the rule base.

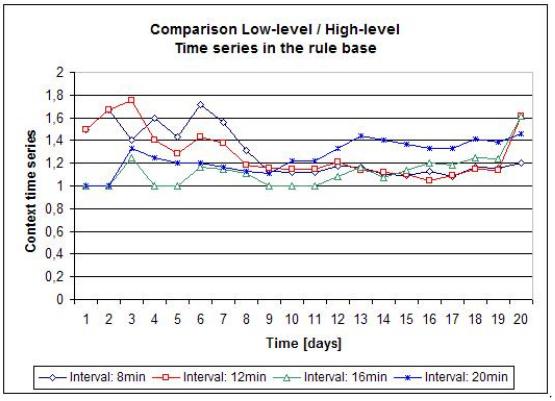


(b) Context elements per time series.

Figure 5.44: Memory consumption of the high-level and low-level context prediction schemes.



(a) Count of rule base entries per observed context element.



(b) Count of time series in the rule base.

Figure 5.45: Memory consumption of the high-level and low-level context prediction schemes.

served that the high-level context prediction scheme has a decreased memory consumption and processing load. For the low-level context prediction scheme, the number of contexts stored in the rule base per observed context is higher. Additionally, the different time series contained in the rule base is increased for the low-level context prediction scheme and the context time series stored in the rule base are longer in case of the low-level context prediction scheme.

Influence of the context interpretation error

We also study the learning and long-term influence of context prediction schemes on the prediction accuracy in a simulation on synthetic input data. We especially investigate the interdependence between the context interpretation error and the percentage of wrong predictions with respect to the simulation time. We utilise the same simulation environment as described in section 5.1.3 and utilise the learning module of the algorithm. An observed time series that is sufficiently different to all recently observed time series in the rule base is added to the rule base. We modulate the adaptability of the algorithm by a learning threshold that describes the adaptability of the algorithm to unknown context patterns. With a learning threshold of τ the algorithm adds time series with $\tau\%$ different context elements to the rule base.

In our scenario, we utilise only one context pattern consisting of 41 different elements. The algorithm is initially trained to this unique pattern. Without any interpretation error all predictions are consequently correct. In various simulation runs, different interpretation errors are applied. Since new context time series are only added to the rule base if a sufficiently large number of errors is contained, larger interpretation errors have a more serious effect on the prediction accuracy. In different simulation runs, the learning threshold of the algorithm is varied. Starting with the perfectly trained prediction algorithm, we feed the pattern 100 times into the algorithm and monitor the contents of the rule base and the prediction accuracy for various interpretation errors and learning thresholds.

Since the interpretation error induces errors in the high-level context time series, we expect the high-level learning procedure to add erroneous patterns to the rule base that might in further predictions decrease the prediction accuracy.

Interpretation error	0.1	0.2	0.3	0.4
Iterations				
10	0.11	0.22	0.29	0.43
30	0.11	0.24	0.32	0.5
50	0.14	0.23	0.32	0.6
100	0.14	0.24	0.29	0.62

Table 5.7: Ratio of erroneously predicted contexts to all predicted contexts with a learning threshold of 0.1 for various interpretation errors.

Interpretation error	0.1	0.2	0.3	0.4
Iterations				
10	6	10	10	10
30	11	29	30	30
50	21	49	50	50
100	38	94	100	100

Table 5.8: Number of rule base entries with a learning rate of 0.1.

Simulation results We first consider the ratio of erroneously predicted context elements to the overall number of predicted context elements to the overall number of predicted context elements. The results with a learning threshold of 0.1 are illustrated in Table 5.7. The number of time series in the rule base is illustrated in Table 5.8.

Note that the ratio of erroneously predicted context elements is typically higher than the context interpretation error. This is due to the fact that incorrect time series are added to the rule base. If the algorithm decides to base its prediction on one of these time series, many of the predicted context elements are to be erroneous. In this way, the interpretation errors are multiplied by the learning method. Since the initial and correct time series is not deleted from the context history, all but one time series in the rule base are incorrect. We observe that the ratio of erroneously predicted context elements is not dependent on the number of erroneous time series

Interpretation error	0.1	0.2	0.3	0.4
Iterations				
10	0.0	0.0	0.24	0.46
30	0.0	0.0	0.36	0.64
50	0.0	0.0	0.42	0.54
100	0.0	0.0	0.41	0.62

Table 5.9: Percentage of erroneously predicted context elements with a learning threshold of 0.3.

in the rule base. A single erroneous time series in the rule base has already significant influence on the context prediction accuracy.

We observe that when we increase the learning threshold of the algorithm, the context prediction errors are bounded to some extent. As illustrated in Table 5.9 we observe no context prediction errors if the context interpretation error is bounded from above by 0.2. With higher context interpretation errors the ratio of erroneously predicted context elements quickly rises to the same values as in Table 5.7. The reason for this is that the number of erroneous context elements in time series in the rule base is relevant to the context prediction error, not the learning threshold of the algorithm.

To further stress this point, we increase the learning threshold to 0.5. The results of this simulation run are illustrated in Table 5.10. Observe that the algorithm is now hardly capable of learning new context time series at all, which consequently improves the prediction accuracy since learning in our case would induce errors into the rule base. Obviously, we do not experience a context prediction error of the algorithm for a context interpretation error up to 0.3. Only with a context interpretation error of 0.4 and after a considerable number of iterations do we observe context prediction errors. With only one incorrect context time series in the rule base, the ratio of erroneously predicted context elements is at 0.61 right from the start. A few incorrect time series in the rule base can therefore have a great impact on the context prediction accuracy.

To summarise, we have observed that the interpretation error has a serious impact on the learning capability of a context prediction algorithm.

Interpretation error	0.1	0.2	0.3	0.4
Iterations				
10	0.0 (1)	0.0 (1)	0.0 (1)	0.0 (1)
30	0.0 (1)	0.0 (1)	0.0 (1)	0.0 (1)
50	0.0 (1)	0.0 (1)	0.0 (1)	0.0 (1)
100	0.0 (1)	0.0 (1)	0.0 (1)	0.61 (2)

Table 5.10: Ratio of erroneously predicted contexts (rule base size) with a learning threshold of 0.5 for various interpretation errors.

The accuracy is decreased due to learning, since errors in observed time series are remembered and again predicted by the context prediction algorithm. Most seriously even, a single erroneous context time series learnt has already a serious impact on the overall context prediction accuracy.

5.2.3 Discussion

In this section we have studied the impact of the context abstraction level on the learning capability of context prediction algorithms.

Our analytic study conducted in section 5.2.1 demonstrated that a high-level context prediction scheme is likely to achieve worse context prediction accuracy than a low-level context prediction scheme in an ideal case.

Furthermore, in simulations on real and synthetic data sets we have been able to confirm this observation. In particular, we showed that the high-level context prediction scheme is disadvantaged by the learning approach, while the low-level context prediction scheme profits from the same approach. In addition, we demonstrated in a simulation on synthetic data that this effect is strongly dependent on the interpretation error for the high-level context time series. Due to errors in the input sequence, the learning approach adapts to erroneous context sequences. The corresponding prediction accuracy is decreased even below the interpretation error probability.

However, regarding the memory and processing consumption we have observed in the simulation on realistic data that the high-level context prediction scheme is more frugal.

5.3 Summary

In this chapter we have studied the context prediction accuracy of low-level and high-level context prediction schemes, as well as different learning capabilities of high-level and low-level context prediction schemes.

Both have been studied analytically first and then in simulations on real and synthetic data. For the context prediction accuracy, several aspects have been identified and discussed. These are the alternative order of context processing steps applied and the different search space sizes for low-level and high-level context prediction schemes.

For the first aspect of the context prediction accuracy, we have observed several benefits of the low-level context prediction scheme over the high-level context prediction scheme. In the analytical discussion we could see that, from the context history and the prediction horizon, an increase in these parameters has less serious effect on the low-level context prediction scheme than on the high-level context prediction scheme. The same is true for the context acquisition accuracy where a decreasing context acquisition accuracy translates to an overall decrease of the context prediction accuracy. This effect is more serious for high-level context prediction than for low-level context prediction.

Most interestingly, the number of high-level and low-level context values in the simulation environment has only little effect on the overall context prediction accuracy for high-level and low-level context prediction schemes. These results could also be observed in the simulations that have been conducted.

The impact of the search space in the context prediction accuracy is twofold.

On the one hand, the information content of the low-level context search space, which translates to benefits for the low-level context prediction scheme, is increased. Consequently, the low-level context prediction scheme is capable of distinguishing general trends in the time series earlier in time. The reason for this is that it is capable to extract more information out of a fixed length context time series. This effect could also be observed in simulations conducted.

On the other hand, the high-level context time series is more robust to measurement errors and possibly provides benefits in scenarios that are susceptible to errors and that do not hinge on trends in observed time series.

From the study of the learning task, we have again observed that the learning method of the high-level context prediction scheme is disadvantaged compared to the low-level context prediction scheme, since the higher number of errors in the input sequence decreases the learning efficiency. This result from the analytic consideration could also be shown in the simulations we conducted.

On the other hand we observed in a simulation on sampled location data that the memory consumption of the low-level context prediction scheme is higher than the memory consumption of the high-level context prediction scheme. We identified the different level of abstraction to account for this property. Several low-level context time series might be represented by a single high-level context time series.

While the low-level context prediction scheme might distinguish between two only slightly different time series, a distinction is not possible in case of the high-level context time series. Consequently, while the accuracy of the low-level context prediction scheme benefits from this property, its memory consumption increases, compared to the high-level context prediction scheme.

6 Context prediction algorithms

> *To me, the primary motivation behind the information appliance is clear: simplicity. Design the tool to fit the task so well that the tool becomes a part of the task, feeling like a natural extension of the person. This is the essence to the information appliance*
>
> (Donald A. Norman: The Invisible Computer [131])

One important piece in an architecture for context prediction is an algorithms well-suited for the context prediction task. Since context prediction in UbiComp environments covers such a huge field of applications and environmental settings, it seems unfeasible to provide a general all-purpose approach that fits all possible requirements best. Depending on the scenario and the prediction aim at hand, researchers therefore apply domain specific approaches.

Consequently, the prediction task in context-aware and ubiquitous computing is accomplished by various algorithms. In analogy to the discussion in [132] for evolutionary algorithms, we can also argue in our case that no general purpose algorithm exists that is on average well-suited to all possible problem domains. We, however, study several algorithms that have been proposed for context prediction tasks for their capability to fulfil requirements that we believe are typical in many context prediction scenarios.

Naturally, requirements for context prediction algorithms are a high prediction accuracy with high prediction horizons at the same time. Furthermore, to be suitable for mobile environments, processing and memory

requirements have to be low. Additionally, a reasonable error tolerance for the input data is desirable. These are some of the most obvious requirements for context prediction algorithms in mobile ubiquitous environments. In section 6.1 we discuss these requirements in more detail.

We then introduce various algorithms that can be utilised for context prediction tasks. The algorithms considered are Bayesian networks, support vector machines or kernel machines. Furthermore, more symbolic approaches as Markov models and neural nets are considered. We also consider nearest neighbour class prediction approaches as the self organising map algorithm, pattern matching and alignment prediction approaches. As representatives of statistic prediction methods we consider the ARMA approach as well as autoregressive and moving average approaches as well as a Kalman filter approach.

The algorithms are analysed for their strengths and weaknesses regarding the requirements stated. Since several of these requirements run counter to others, we expect context prediction algorithms to perform well in some disciplines, while in others they are outperformed by competing prediction algorithms. In the discussion of suitable algorithms, we face the task of finding the algorithm that constitutes the best compromise to all requirements. Through this discussion we identify a set of algorithms that are best suited for mobile ubiquitous environments.

These algorithms are then further applied to real context prediction scenarios. In these scenarios we are especially interested in the context prediction accuracy of the algorithms, in the prediction accuracy with respect to an increasing context prediction horizon, and in algorithm specific features of the data sets. The scenarios are chosen from different, unrelated application domains so that the reader can get an impression of the domain independent capabilities of the prediction algorithms.

In conclusion to this discussion we recommend context prediction algorithms for typical context prediction scenarios.

6.1 Aspects of context prediction algorithms

This section discusses several aspects of context prediction algorithms. This list of aspects is definitely not complete but rather summarises the features of context prediction algorithms that we believe are most crucial in ubiquitous computing scenarios.

6.1.1 Prediction accuracy

The problem of context prediction is essentially an optimisation problem. The search space is composed of all time series that can possibly be predicted. The context prediction algorithm has then to find the time series in the search space that will actually be observed in the future. If prediction errors occur, they are to be minimised.

Probably the two most basic features that constitute a good context prediction algorithm are a low error probability combined with a high prediction horizon. We are interested in algorithms that routinely compute the predicted time series with high accuracy in the prediction and that maximise the prediction horizon at the same time.

The speed with which the prediction accuracy decreases with increasing prediction horizon might further matter for the comparison of context prediction algorithms. The basic question is, how many future contexts can be predicted by the algorithm and, even more important, how fast does the prediction accuracy decrease with an increasing prediction horizon.

6.1.2 Adaptability

Another aspect is the adaptability or learning time of the algorithm to new or changing environments. New user habits or a changed context evolution process might inflict changes to the observed context patterns. We generally assume that these typical patterns are of a constantly changing nature. Consequently, old context patterns become outdated and new typical context patterns are observed. In order to provide a maximally accurate prediction accuracy it is crucial that the set of typical context patterns is up to date at all times. Possible solutions to this challenge are flexible weights assigned to typical patterns or a constant learning procedure. Lower weighted or less important context patterns can then be removed from the set of typical context patterns.

6.1.3 Memory and processing load

We assume that mobile, personal devices are the main target platform for context prediction approaches. These devices are typically restricted in memory resources or processing capabilities. The memory requirements for a context prediction method shall therefore be as low as possible. Considering this aspect, a good context prediction approach is able to achieve a

fixed prediction accuracy with minimum memory requirements. Or stated otherwise: which algorithm scores the best prediction accuracy with fixed memory usage requirements. The same challenge does generally apply for the processing load.

An interesting approach to solve the memory and processing restrictions is the utilisation of remote processing power or remote storage. In our work we do not consider the possibility of external storage or processing resources.

6.1.4 Multi-dimensional time series

When several context sources are acquired by the context prediction algorithm, a prediction for a multi-dimensional context time series is to be provided. Not all algorithms are applicable to multi-dimensional time series naturally. However, by combining the context time series elements of multi-dimensional context time series to a one-dimensional time series of condensed, more expressive context time series elements, the observed time series is trivially transformed to a one-dimensional context time series. Alternatively, various one-dimensional context time series might be utilised instead of a single multi-dimensional time series. An algorithm that is only applicable to one-dimensional time series might then be applied multiple times to the distinct one-dimensional time series. However, with this approach, interrelations between the distinct one-dimensional context time series that might also provide valuable information for context processing can then not be considered by the context processing algorithms.

6.1.5 Iterative prediction

For context prediction we are typically not only interested in one or the most recent future context time series element, but in complete context time series of greater length. The prediction of a complete future context time series in one step is not possible for various prediction algorithms. These algorithms typically iterate the prediction process several times with already predicted data.

The drawback of this procedure is that predicted context values are naturally of less credibility than observed context values, since the iterated prediction is based on already predicted context elements that are typically more error prone than observed context elements. Therefore, the

time series data a prediction is based on becomes less reliable the longer the prediction horizon becomes. We consequently expect that for an iterative prediction process, the accuracy of context objects that are predicted further away in the prediction horizon quickly diminishes.

6.1.6 Prediction of context durations

Various prediction algorithms might have different prediction capabilities. Some algorithms are, for example, capable of predicting a context object accompanied with a probable time stamp, while others are not capable of predicting time stamps. We are not only interested in the typical contexts that will occur in the future but also in the order and occurrence time or duration of the predicted contexts. We therefore consider the ability to predict the occurrence or the time stamp of a given context as one crucial feature of a context prediction algorithm.

For some algorithms that do not naturally support the prediction of time stamps together with the predicted contexts, an alternative approach might be applied. When considering the time samples also as context time series and composing a new context time series of increased dimension that also includes the time, a timestamp for the predicted context elements is easily achieved, provided the algorithm is applicable to multi-dimensional and possibly multi-type context time series data.

6.1.7 Relaxation of typical behaviour patterns

Context sequences that describe human behaviour patterns implicitly contain inaccurate context elements. The capability to tolerate inaccurate measurements, or even unpredictable changes in time series data to some extent, is vital for a context prediction algorithm in order to achieve a high prediction accuracy. An unexpected call, for example, might change the user context momentarily. After the completion of the call however, the user might continue with the context pattern she initially started with. Of even greater significance, since context prediction in ubiquitous computing environments might be based on behaviour patterns of humans instead of execution patterns of deterministic machines, we must expect that even for repetitions of behaviour patterns, context durations gradually change.

However, the likelihood of the prediction algorithm to deviate from exact matching patterns must not decrease the prediction accuracy. The

algorithm has to identify important contexts and context changes and has to abstract from less important ones.

6.1.8 Context data types

The number of context processing algorithms applicable to a context prediction scenario varies depending on the context data types apparent in the scenario. However, the more complex the data types in a scenario, the more ambiguous context prediction methods might be applied. context of more complex data types contain additional information that might be useful for the prediction task. Consider, for example, numerical contexts and ordinal contexts. While all operation applicable to ordinal contexts are also applicable to numerical contexts, the opposite is not true. Trends in context time series might, for example, be observed for numerical context time series but not for ordinal context time series.

Popular context prediction methods implicitly support nominal contexts but do not profit from contexts that provide additional information. Although several algorithms are applicable to hierarchical contexts, we do not know of any prediction algorithm that makes use of the hierarchical property of contexts in a sense that the knowledge about these hierarchy improves the prediction accuracy (Even though this would constitute an intriguing research question). Table 6.1 classifies the algorithms that are introduced in section 6.2 according to their ability to nominal, ordinal, hierarchical or numerical contexts. Since numerical contexts summarise all different operations, this context type can be handled by all algorithms proposed.

Furthermore, for multi-dimensional time series, the context data types of the different dimensions might differ in one time series. In order to be applicable to arbitrary context time series, a context prediction algorithm has to provide a prediction method that is ignorant of the context data types.

6.1.9 Pre-processing of time series data

Some context prediction algorithms require patterns of typical contexts preparatory to their prediction task. From these typical time series, these algorithms can identify similar patterns in the observed time series. Typically, this pre-processing process constitutes the most important part of

Algorithm	Ordinal contexts	Nominal contexts	Hierarchical contexts	Numerical contexts
BN[1]	+	+	+	+
SVM[2]	-	-	-	+
KM[3]	-	-	-	+
MM[4]	+	+	+	+
NN[5]	+	+	+	+
NNS[6]	-	(+)[7]	(+)[7]	+
SOM[8]	-	(+)[7]	(+)[7]	+
PM[9]	+	+	+	+
AP[10]	(+)[7]	(+)[7]	(+)[7]	+
ARMA	-	-	-	+
AR	-	-	-	+
MA	-	-	-	+
Kalman filters	-	-	-	+

Table 6.1: Context data types applicable to various algorithms.

[1] Bayesian Networks
[2] Support Vector Machines
[3] Kernel Machines
[4] Markov Models
[5] Neural Nets
[6] Nearest Neighbour Search
[7] Provided a suitable distance metric
[8] Self Organising Map
[9] Pattern Matching
[10] Alignment prediction

the context prediction task. Appropriate pre-processing can increase the prediction accuracy and also the prediction speed [133].

Typical context patterns have to be extracted from data samples of the observed context time series. Characteristic parts of these time series are identified while the noise, ie irrelevant contexts in the input data are removed. Algorithms for pre-processing time series data are for instance introduced in [87, 107]. The pre-processed data of these algorithms can be utilised for various context prediction methods. However, this pre-processing requires additional processing power. Prediction methods that do not require any pre-processing of input data might therefore be less computationally complex.

6.2 Context prediction methods

This section introduces several algorithms that are suitable for context prediction tasks. For each algorithm presented, we describe the general idea and the way in which it might be applied to context prediction tasks. In conclusion, we obtain strengths and weaknesses of each algorithm which enables us to figure out those algorithms that are best suited for context prediction tasks.

6.2.1 Support Vector Machines

A support vector machine (SVM) is a method to solve search problems in multi-dimensional vector spaces. SVMs have been studied already in the late seventies [134]. They are, however, receiving increased attention only in recent years. The basic idea is to find a geometrically interpretable separator for the input data that enables the separation of the data into several classes [135, 136].

Introduction to SVMs

Suppose, we are given training data

$$(\overrightarrow{x_{t_0-k}}, y_{t_0-k}), \ldots, (\overrightarrow{x_{t_0}}, y_{t_0}) \tag{6.1}$$

where the $\overrightarrow{x_i}$ represent vectors of an Euclidean vector space and $y_i \in \{-1, 1\}$ represents a label for vector i. These labels define the class a vector $\overrightarrow{x_i}$ belongs to. From this training set, the task is to find some rule

that enables the distinction of further observations $\overrightarrow{x_j}, j = t_{0+1}, \ldots, t_{0+n}$ into any of the two classes 1 or -1 with low probability of error.

Since the $\overrightarrow{x_i}$ are vectors they can be represented together in a corresponding vector space, the input space. In case the training points are linearly separable with respect to the labels 1 and -1 by a hyperplane in this vector space, the task is to find the hyperplane that maximises the distance to those vectors that have the minimum Euclidean distance to the hyperplane. These nearest points are called the support vectors, since the optimal separating hyperplane would be identical even if all vectors short of the support vectors were removed from the training set.

A support vector machine consequently is an algorithm that finds this optimal separating hyperplane given a set of training points.

In case of input vectors that are linearly not separable in the input space, the restriction given by the hyperplane is relaxed to some extent. The requirements for the points to be situated at one side of a hyperplane can be expressed by a system of inequalities. To relax the strict bounding by the hyperplane, so called 'slack-variables' are included into the system of inequalities that then allow some vectors to be positioned on the 'wrong' side of the hyperplane.

In the previous discussion we have restricted the number of classes that may be distinguished between a SVM to two, namely -1 and 1). A possibility to generalise the number of classes to r is to divide the problem into r two-class problems that are subsequently solved for any single input vector v_j. In each of these two-class problems we test, if the vector v_j belongs to one of the r classes or not. In r iterations the class of the vector can thus be determined.

In the discussion above we have assumed linear SVMs, ie a search space that is linearly separable. The SVM-approach can easily be extended to the non-linear case by the utilisation of a transformation function that maps all points in the search space into a higher-dimensional feature space in which the points are again linearly separable. If data is mapped into a space of sufficiently high dimension, it is always linearly separable [137]. This mapping is represented by a so-called kernel function. Depending on the specific problem encountered, different kernel functions might be sufficient. The main difficulty is then constituted by the search for the optimal kernel function for one problem domain.

Context prediction with SVMs

For context prediction, time series of context elements constitute the input of the prediction algorithm. Since context elements can be represented as vectors, and time series of context elements as still higher-dimensional vectors that are simply concatenated from the vectors representing context elements, the adaptation of SVMs to context prediction is quite natural. The input vectors $\overrightarrow{x_i}$ represent time series of context elements. Every class the SVM is capable of distinguishing between represents a typical time series. The SVM can then – given the observed time series – compute the typical time series it is most similar to and provide the continuation of this time series as output for the prediction.

Note that this operation requires the knowledge of all typical time series as input for the SVM prediction algorithm. A pre-processing of context elements is therefore required in order to figure out the typical context time series. This pre-processing is to be applied in addition to the training of the SVM to typical time series that define the separating hyperplanes.

Adaptive operation of the SVM predictor therefore requires an iterative process in which the pre-processing and training phases are constantly applied to newly observed context sequences.

Discussion

Support vector machines provide a comparably simple approach to relate a pattern to one of a finite number of classes and additionally provide a simple geometric representation for the search problem. The approach is especially suited to handle problems with a high number of input dimensions. Additionally, the support vector machine definitely finds a global minimum to the possibly multidimensional search problem.

However, the contents of the recognised patterns are restricted to a numerical representation. Moreover, the choice of the best suited kernel is still an open research issue for SVMs. Another, most serious issue when it comes to context prediction, is that discrete data presents a problem for support vector machines [136]. For the applications to context prediction (or time series prediction in general) a further limitation arises due to the fixed size of the input space. Only patterns of exact identical length can be compared by a fixed SVM. Since typical context patterns are not necessarily of equal length, this property hinders a successful adaptation of SVMs to context prediction tasks.

Finally, the adaptation of a SVM-predictor to a changing environment is possible but computationally expensive since pre-processing operations have to be constantly applied. The runtime of the method depends primarily on the time to compute the separating hyperplane. Assume that the number of training sequences utilised for creating the separating hyperplanes is ς. The number of training points is then given by $k\varsigma$. Empirical studies suggest an average training time of $O(k\varsigma) = O(k)$ [138, 139]. The process has to be repeated constantly since the UbiComp environment is expected to frequently change.

6.2.2 Markov models

Markov processes are intensively studied and constitute a major branch in the theory of stochastic processes. Due to an intuitive graphical representation, it is possible to illustrate some properties of stochastic processes graphically. Markov models are popular also for their simplicity and easy applicability to a huge set of problems in various domains. A comprehensive description of Markov processes can be found in [140].

Introduction to Markov processes

Assume a set of possible events $\chi_1, \ldots, \chi_n$ of a stochastic process. In modelling this process by a Markov chain, we assume that any event χ_i is exclusively dependent on the event directly preceding χ_i. This lack of memory in a stochastic process is called the Markov property. Typically, we speak of the Markov property, if the probability distribution of any state of a stochastic process is exclusively dependent on the current state of the process and not on any prior states.

However, it is also possible to define similar processes in which any event χ_i is exclusively dependent on a finite set of preceding events $\chi_{i-k}, \ldots, \chi_{i-1}$. In this case, we speak of higher order Markov chains. If the states of the statistical process is dependent on χ_i and $k-1$ preceding events, we speak of an order-k Markov process.

This process might be illustrated as a directed labelled graph $G = (V, E)$ with states (or vertices) in V. Transition probabilities between $\nu_i, \nu_j \in V$ are represented by the labelling of the edge $\overline{\nu_i \nu_j} \in E$.

Context prediction with Markov processes

Given a sequence of context time series elements $\xi_{0-k+1}, \ldots, \xi_0$, we can easily generate a Markov model representing the transition probabilities for each pair of these observations. Provided with such a Markov model and the current context time series element ξ_0, we are able to provide a probability distribution on the next outcome. Note that this can also be generalised to higher order Markov processes. Several iterations of this process might also provide predictions with higher prediction horizons.

Discussion

The Markov model is straightforward and easily applied to context prediction tasks. The model can be applied to numerical and non-numerical data alike. However, a prediction that reaches farther into the future implicitly utilises already predicted data which might consequently decrease the prediction accuracy (cf. section 6.1.5).

The runtime of this approach is dependent on the size of the probability graph G. Each context time series element is considered a state in a Markov chain. Let C be the set of different contexts and context values observed by the algorithm. The number of states of the Markov chain is then consequently C. The arcs between the states describe the probability to traverse from one state to another. The future states with highest probability are desired. The time to find the most probable next state is $O(|C|)$ in the worst case. This is because every arc to a possible following context time series element has to be considered and $|C| - 1$ arcs are existent in the worst case. To find the most probable n future context time series elements, the naive computation time is $O(|C|^n)$. However, when the transition probabilities are stored to a matrix, one matrix multiplication for every one future context suffices. The computation time can consequently be approximated by $O(n \cdot |C|^2)$.

6.2.3 The state predictor method

The state predictor method is a comparably new prediction approach that has been developed by Jan Petzold at the University of Augsburg [80, 81, 82, 83, 63, 84]. This prediction approach constitutes a simple prediction tool with low processing requirements. It has its origin in branch prediction techniques that are also applied in current microprocessors [79].

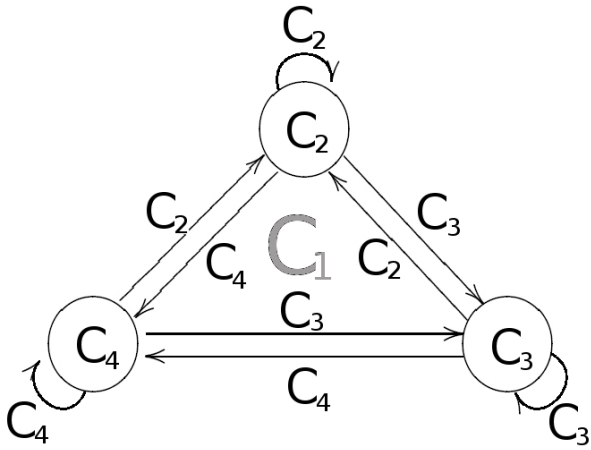


Figure 6.1: The state predictor.

In a graphical representation, the state predictor might be depicted as a state machine.

Context prediction with state predictor methods

Assume a fixed set of contexts $C = C_{0-k+1}, \ldots, C_0$. For every one context $C_i \in C, i \in 1, \ldots, k$ a state machine is constructed from the contexts $C_j \in C, j \neq i$ in which every state represents one context from the context set short of C_i. Between every pair of states a transition is possible.

The general idea of the state predictor method is illustrated in figure 6.1 for a context set $C = \{C_1, C_2, C_3, C_4\}$. The state machine depicted in the figure indicates, which context is likely to be observed next. The current state of the state machine represents the context that will be predicted. If another context than the predicted one actually occurs, the state of the state machine is updated to that state which represents the observed context. The state machine in this way always stores the context transition that was observed for the last transition made.

Assume, for example, that the current context is C_1 and the state machine associated with context C_1 is the one illustrated in figure 6.1. If the

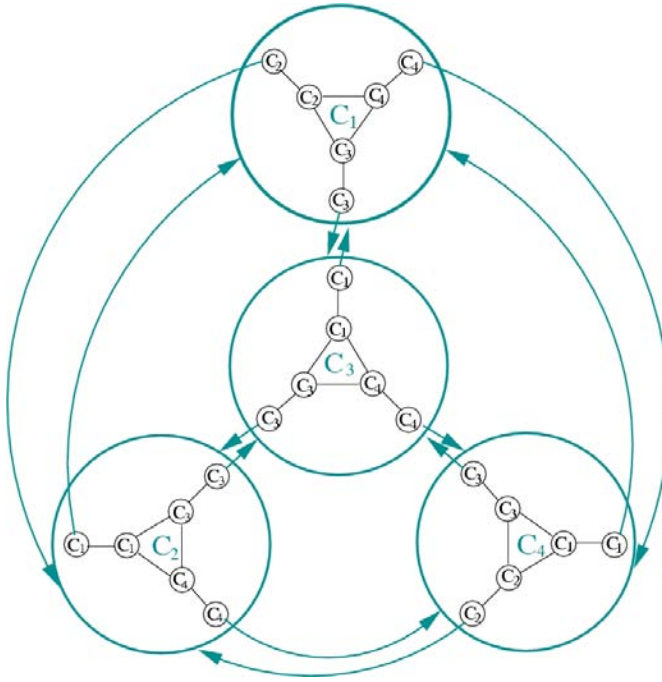


Figure 6.2: The state predictor embedded into a Markov model.

state machine is currently in state C_2, for instance, then C_2 is considered the next observed context. Should, however, the next context actually be C_3, the current state of the state machine changes to C_3 and C_3 is consequently predicted to be the future state the next time when the context is C_1.

Therefore, with every context, a different state machine is associated. If we connect these state machines we obtain a view on the whole picture as it is schematically illustrated in figure 6.2.

Note that this is basically a simple Markov model where the transition probabilities are only allowed to take the values 0 or 1.

Discussion

The state predictor approach combines low memory and processing requirements with an intuitive graph representation. It is, however, only applicable to symbolic context elements. Basically, the method is able to predict the next context transition but is not able to predict context durations.

Furthermore, it is suitable for one-dimensional context time series only. A simple learning capability is implicitly implemented by the state predictor method. This, however, is only capable of storing the last observed context pattern.

Further work by the University of Augsburg proposes some improvements to state predictors. If we keep our focus on the underlying Markov chain, we observe that the proposed improvements can still be modelled by a Markov model, while only the mechanism to update the transition probabilities changes. Only transition probabilities with values 1 or 0 are allowed in all proposed modifications.

For every context predicted, all possible states have to be checked for their value (0 or 1). In order to find the state with value 1, $O(|C|)$ states have to be considered in the worst case. To find the most probable n future context time series elements, the computation time is $O(|C|^n)$. In analogy to the Markov prediction method, the computation time can be improved to $O(k|C|^2)$.

6.2.4 Self organising maps

The self organising map (SOM) algorithm was proposed by Teuvo Kohonen [141, 142, 143, 144]. The author presented it as a model of the self-organisation of neural connections. A self organising map is a topology preserving lattice of a predefined number of nodes that represents a topology of elements in the input space.

The Algorithm inherits the self-organisation property, which means that it is able to produce organisation starting from possibly total disorder. In the SOM algorithm, a neighbourhood structure is defined and preserved between all nodes of the map in the course of the execution.

Generally, the set of elements from the input space are grouped into subsets, each of which is represented by one node of the lattice. Consequently, the lattice defines a neighbourhood between these subsets. A representative or prototype can be defined for each subset.

Introduction to self organising maps

We recapitulate a condensed definition of the SOM algorithm that can be found in [145].

Let $I = \{\overrightarrow{\eta_1}, \ldots, \overrightarrow{\eta_{|S'|}}\}$ be a set of km-dimensional vectors that are associated with nodes in a lattice. The neighbourhood structure is provided by a symmetrical, non-increasing neighbourhood function $d : I \times I \to \mathbb{R}$ which depends on the distance between two nodes $\overrightarrow{\eta_i}$ and $\overrightarrow{\eta_j}$ for $\overrightarrow{\eta_i}, \overrightarrow{\eta_j} \in I$. The input space S is a bounded subset of $\mathbb{R}^{km}$, where $km \in \mathbb{N}$ is the dimension of the input vectors $\overrightarrow{v(t)} \in S$. The state of the map at time t is given by

$$\eta(t) = \left(\overrightarrow{\eta_1(t)}, \overrightarrow{\eta_2(t)}, \ldots, \overrightarrow{\eta_{|S'|}(t)}\right), \tag{6.2}$$

The SOM algorithm is recursively defined by

$$\begin{aligned} i_c\left(\overrightarrow{v(t+1)}, \overrightarrow{\eta(t)}\right) &= argmin\left\{\left\|\overrightarrow{v(t+1)} - \overrightarrow{\eta_i(t)}\right\|, \overrightarrow{\eta_i(t)} \in \eta(t)\right\}, \\ \overrightarrow{\eta_i(t+1)} &= \overrightarrow{\eta_i(t)} - \varepsilon_t d\left[i_c\left(\overrightarrow{v(t+1)}, \overrightarrow{\eta(t)}\right), \overrightarrow{\eta_i}\right] \\ &\quad \cdot \left(\overrightarrow{\eta_i(t)} - \overrightarrow{v(t+1)}\right), \forall \overrightarrow{\eta_i} \in I. \end{aligned} \tag{6.3}$$

In this formula, $i_c\left(\overrightarrow{v(t+1)}, \overrightarrow{\eta(t)}\right)$ corresponds to the node in the network that is closest to the input vector. The parameter ε_t controls the adaptability of the self organising map.

Context prediction with self organising maps

The process of predicting with SOMs is divided into two stages. The preparation and clustering phase in which the model utilised for the prediction is created and the actual prediction stage. In the following sections these two stages are discussed before the benefits and drawbacks of the SOM-prediction approach are discussed. Recent work on the utilisation of SOM algorithms for prediction tasks can be found in [90, 78, 92, 93, 94, 95, 146].

Model creation phase In the modelling phase for any $q, i, j \in \mathbb{N}$, the context history is divided into the set of all time series of length q. Let $T_{i,j}$ be the context history observed by the algorithm. For $r \in [0, \ldots, q]$ the set $T_{i+r,j-q+r}$ of time series of length $j - i + q$ is created for $T_{i,j}$. In

addition, for $t \in [i+r, \ldots, j-q+r]$, so-called deformations $D_{t,t+r}$ are created with

$$D_{t,t+r} = T_{t+1,t+j-i+q+1} - T_{t,t+j-i+q}. \tag{6.4}$$

These deformations are time series that describe the modifications necessary in each time series element to evolve from one time series to the time series occurring one time instant later.

These sets of time series $T_{i+r,j-q+r}$ and $D_{t,t+r}$ are both clustered with help of the vector quantisation by SOM. The outcome of this procedure are for $i, j \in \mathbb{N}$ prototypes $\overline{T_i}$ and $\overline{D_i}$ of time series that represent a set of similar time series and deformations respectively. Finally, a matrix M is created that describes the relation between the $T_{t,t+r}$ and $D_{t,t+r}$ time series.

For a fixed i and $j \in [1, \ldots, \kappa]$ with κ describing the number of different prototypes $\overline{D_i}$, the row M_{ij} represents the conditional probability that $D_{t,t+r}$ belongs to $\overline{D_j}$ given that $T_{t,t+r}$ belongs to $\overline{T_i}$.

Prediction phase When these preparations are made, the prediction consists of seven steps.

1. For any time t, consider a time series $T_{t-q,t}$.
2. Find the associated prototype.
3. Randomly choose a deformation $\overline{D_j}$ according to the probability distribution given by M.
4. Obtain the prediction for time $t+1$ as $T_{t-q+1,t+1} = T_{t-q,t} + \overline{D_j}$.
5. Iterate these steps with the obtained predicted time series until the prediction horizon is of the desired length.
6. A Monte-Carlo procedure is used to repeat this process several times.
7. From all obtained predictions, extract global trends of the time series as the evolution of the time series mean, its variance or confidence intervals.

In [90], a procedure is presented with which whole future time series can be predicted in one step instead of iterating the one-step-ahead prediction

several times. Basically, the authors propose to utilise vectors of consecutive observations in the to be clustered time series, instead of single observations repeatedly. Consequently, a vector of consecutive observations is then predicted instead of a single time series element.

Discussion

A benefit of this prediction method is that it implicitly groups similar time series together and represents them by a prototype. A separate pre-processing algorithm in order to detect typical time series patterns is therefore not required. As described in section 3.2.4, we expect the context prediction algorithm to observe typical patterns in the context history. These typical patterns will likely reappear only in similar, but seldom in an identical, fashion. The grouping of similar time series to prototypes is therefore well-suited for context prediction tasks. Further benefits are the implicit utilisation of several statistical measures.

However, since the method utilises predicted, possibly error prone contexts for the prediction of horizons that exceed 1, the prediction accuracy is expected to decrease quickly with increasing prediction horizon.

The runtime of the algorithm to state one prediction can be estimated as $O(|S'|)$ since for a given input vector all $|S'|$ vectors in the lattice are compared for their distance to the input vector. When the input vectors that describe the lattice are ordered in a tree structure, the runtime for one prediction can even be reduced to $O(log(|S'|))$. However, the model creation phase is not considered in this calculation. In changing environments, the model creation phase is to be repeatedly applied in regular intervals. The model creation phase requires an additional time of $O(|S'|^2)$, the maximum number of time series considered for the model creation phase. When we estimate the number $|S'|$ of input time series by $\varsigma \cdot k$ for a suitable constant $\varsigma \in \mathbb{N}$, we obtain $O(k^2)$ for the overall runtime of the algorithm.

More seriously even, the adaptability of the approach is low and restricted to the level of adaptability the application designer permits. The number of different typical context patterns, ie prototype vectors is restricted by the fixed size of nodes in the lattice considered.

In the alternative consideration where whole time series are predicted instead of single time series elements, the size of the context history is to be increased dramatically. For prototype vectors of length q and a desired

prediction horizon of length n the context history is required to have length $n \cdot q$ at minimum. However, in this minimum case only one prototype vector would be created. In order to have r prototype vectors, a context history length of $(n + r) \cdot q$ is required at least. We argue that typical context patterns are seldom of extensive length in representative applications. The one step prediction approach is therefore seldom applicable to context prediction tasks.

Another drawback is the implicit restriction to numerical context patterns. While the clustering by the SOM is also applicable to contexts of nominal context data types, the creation of the Matrix M and the creation of the deformation vectors requires a distance metric between all context elements that provides a real valued output when applied to any pair of context elements.

Multidimensional time series are not yet considered for this approach and an extension to the multidimensional case would, while theoretically possible, further inflate the model.

The SOM prediction approach is therefore not well-suited for context prediction tasks.

6.2.5 Pattern matching

Given a sequence of elements and a set of prototype sequences, the idea of pattern matching is to find a substring of the element sequences that matches a sub-sequence in one prototype. The matching can be exact or approximate. Since typical observed context patterns likely contain random fluctuation in the data (cf. section 6.1.7), we argue that exact matching approaches are unsuited for context prediction tasks. In scenarios where the sampled input data might contain errors, exact matches cease to be useful in order to find similarities between data sequences. For approximate matches, several implementations are possible. In the next sections, we introduce the alignment matching as one possible approximate matching method. A related approach is, for example, the ONISI algorithm that has been introduced in [88].

Alignment methods

The idea of the alignment approach is to find in two string sequences the most similar sub-sequence by adding gap-symbols to one or the other observed sequence. A decent introduction to alignment methods is given in

[106]. The following definitions are adopted to our notation.

Definition 6.2.1 : Alphabet

An alphabet Σ is a finite, non-empty set of symbols σ where each symbol uniquely matches a configuration of raw data values of one time interval t_i.

We represent patterns of elements as strings. A formal definition of a string is

Definition 6.2.2 : String

A string s over the alphabet Σ is a finite row of symbols from Σ. The length $|s|$ of a string s is given by the count of symbols in s. The set of all strings of length n over the alphabet Σ is denoted Σ^n. Furthermore, Σ^* denotes the set of arbitrary length strings over the alphabet Σ. With ss' we denote the concatenation of the strings s and s'.

Every context time series $T_{i,i+k} = \xi_1 \dots \xi_k$ is basically also a string of context time series elements $\xi_i \in \Sigma$ where every time series element is described by a symbol $\xi_i \in \Sigma$. In the following discussion we switch to this time series related notation.

Definition 6.2.3 : Substring

For $k, n \in \mathbb{N}$ let $\xi = \xi_1 \dots \xi_k$ and $\xi' = \xi'_1 \dots \xi'_n$ be strings over the alphabet Σ.

- *ξ is a substring of ξ', if two strings ρ and ν exist with $\xi' = \rho\xi\nu$.*
- *ξ is a prefix of ξ', if a string ρ exists with $\xi' = \xi\nu$.*
- *ξ is a suffix of ξ', if a string ρ exists with $\xi' = \rho\xi$.*

Given two strings $\xi = \xi_1 \dots \xi_k$ and $\xi' = \xi'_1 \dots \xi'_n$ the alignment method finds a sequence $\xi_i \dots \xi_j$ which is a substring of ξ with maximum similarity to ξ'. This substring is called an alignment between ξ and ξ'.

Definition 6.2.4 : Alignment

Let $\xi = \xi_1 \dots \xi_k$ and $\xi' = \xi'_1 \dots \xi'_n$ be two time series over the alphabet Σ. And $\{-\} \notin \Sigma$ a gap symbol. Let $\Sigma' = \Sigma \cup \{-\}$ and $h : \Sigma'^* \to \Sigma^*$ a homomorphism with $\forall \sigma \in \Sigma : h(\sigma) = \sigma$ and $h(-) = \lambda$. An alignment of ξ and ξ' is a pair (ρ, ν) of strings of length $l \geq \max\{k, n\}$ over the alphabet Σ' so that the following conditions hold:

1. $|\rho| = |\nu| \geq \max\{|\xi|, |\xi'|\}$
2. $h(\rho) = \xi$
3. $h(\nu) = \xi'$
4. *there is no position where both ρ and ν have a gap.*

To rate the similarity between two strings, a similarity metric defining the similarity between two letters $\sigma, \sigma' \in \Sigma$ is needed.

We refer to the function representing this metric as the alignment rating.

Definition 6.2.5 : Alignment rating

Let $align(\xi, \xi')$ be an alignment of two time series with $\xi = \xi_1, \dots, \xi_k$ and $\xi' = \xi_1, \dots, \xi_n$. An alignment rating $r : \Sigma^* \to \mathbb{R}$ is a metric describing the similarity between the time series ξ and ξ'.

Since we are not interested in the alignment of whole time series, but want to find sub–sequences in time series that are maximally similar according to a given alignment rating, we have to further modify the alignment description given above.

This leads to the following definition.

Definition 6.2.6 : Local alignment

Let r be an alignment rating with the optimisation aim minimisation. A local alignment of two strings $\xi = \xi_1 \dots \xi_k$ and $\xi' = \xi'_1 \dots \xi'_n$ is an alignment of substrings $\rho = \xi_{i_1} \dots \xi_{i_2}$ and $\nu = \xi'_{j_1} \dots \xi'_{j_2}$. An alignment $align(\xi, \xi')$ is an optimal local alignment of ξ and ξ' considering r if

$$r\left(align(\xi, \xi')\right) = \min\{ \quad d(\rho, \nu) \quad | \rho \text{ is a substring of } \xi, \; \nu \text{ is a substring of } \xi' \} \ .$$

In this formula, $d(\rho', \nu')$ is the distance metric of the local alignment.

We are especially interested in local alignments between the end of the observed sequence and an arbitrary position of the typical sequence. This type of alignment is referred to as semiglobal alignment.

Given some distance metric and two time series, we are able to provide a prediction regarding one of these time series by searching for the optimal semiglobal alignment between these time series.

Context prediction with alignment methods

The approach to apply alignment methods to context prediction tasks is a comparably new technique that has been first introduced in [8]. In order to utilise alignment methods for the context prediction task, the method has to be expanded. Alignment methods can be utilised in order to find similarities between time series. However, a prediction is a most probable continuation of an observed context time series.

The general idea is the following. In addition to the standard alignment approach, we require a set of typical context patterns. This set can, for instance, be created by any pre-processing methods as mentioned in section 6.1.9. We refer to this construct as the rule base of the algorithm, since it represents the rules that guide the prediction process.

The rule base contains all typical context time series and provides also all possible predictions. It consequently constitutes the search space of the alignment prediction approach. In analogy to the notion of the search space, we therefore denote the rule base with S.

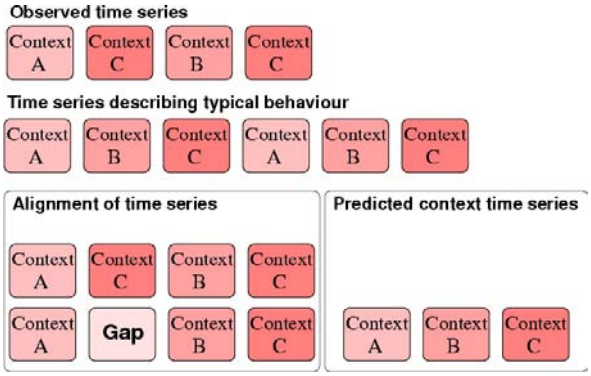


Figure 6.3: Context prediction by alignment methods.

Given a sequence of observed context time series elements $T = \xi_{0-k+1}, \ldots, \xi_0$ we calculate for every typical context time series $T' = \xi'_1, \ldots, \xi'_\kappa \in S$ the optimal semiglobal alignment that matches with the end of the observed time series. The result is a set of optimal semiglobal alignments. Let $\xi_1 \ldots \xi_k$ be the semiglobal alignment with the best alignment rating. The alignment approach then provides the continuation $\xi_{k+1} \ldots \xi_{k+n}$ of this alignment as prediction for the continuation of the observed time series. Figure 6.3 schematically illustrates the approach. For ease of presentation all time series in the figure have dimension one.

In other words, from the set of typical context time series, semiglobal alignments are constructed with the end of the observed time series. The best matching one is chosen for prediction purposes. This typical context time series therefore best approximates the end of the observed time series. Since the aligned time series are most similar at the position aligned and since one of these time series describes a typical behaviour pattern, that regularly occurs, we assume that also the continuation of both time series is most similar to each other. The continuation of this most similar typical time series is consequently chosen for prediction.

The similarity of two time series is the sum over the pairwise similarities of their time series elements.

Let $T = \xi_1, \ldots, \xi_k$ and $T' = \xi'_1, \ldots, \xi'_n$ be two time series with $|T| = k$

and $|T'| = n$. Let S' be the search space of the algorithm and $|S'|$ the size of S'. Since we align two time series $T, T' \in S'$ to each other to find the one subseries in T that has the maximum similarity to a subseries in T', we have to define a similarity rating between two subseries. Assume that ξ_i and ξ_j are two possibly empty time series elements. We describe the similarity between two time series elements by a similarity metric $d : \xi_i \times \xi_j \rightarrow \mathbb{R}$. For $\xi'_i \in T'$ and $\xi_i \in T$, we calculate the similarity between two time series elements at position i in both series as $d(\xi_i, \xi'_i)$. For $\overline{\xi} \in \{T, T'\}$ we define a penalty rating $d(-, \overline{\xi}) = d(\overline{\xi}, -)$. If the algorithm does not align ξ_i and ξ'_i but instead one of the two time series elements to a gap symbol $-$, the cost of this operation is $d(\overline{\xi}, -)$.

The optimum alignment between two time series is found by the Needleman and Wunsch algorithm [127]. A $(k+1) \times (n+1)$ matrix M is created. We initialise the Matrix M by $M_{1,1} = \cdots = M_{k+1,1} = M_{2,1} = \cdots = M_{n+1,1} = 0$. All other entries of M ($i \in \{2, \ldots, k+1\}$, $j \in \{2, \ldots, n+1\}$) are created by integer programming:

$$M_{i,j} = max\{M_{i-1,j-1} + d(\xi_i, \xi'_j); M_{i,j-1} + d(\xi_i, -); M_{i-1,j} + d(-, \xi'_j)\}.$$

Afterwards, the alignment ratings of all possible alignments can be found in row $k+1$ of the matrix M. For a detailed discussion of this method we refer to [106]. Let $\tau_{sim} \in \mathbb{R}$ be a similarity threshold. All alignments with alignment ratings below τ_{sim} are considered important by the algorithm. Let $\xi_i \ldots \xi_j$ be an alignment with an alignment rating below τ_{sim}. The sequence $\xi_{j+1} \ldots \xi_n$ is then chosen as the sequence to be predicted by the alignment prediction method.

Discussion

The alignment prediction approach constitutes a flexible prediction method that is especially well-suited to find typical context patterns in a time series of contexts. The length of the typical context patterns is arbitrary but bounded from above by the context history length. Preparatory to the prediction task, a pre-processing of training data is required in order to provide a set of typical context patterns. This can, however, be implemented in a continuous process that iteratively appends newly observed typical patterns when required. Already computed context patterns remain valid.

For the estimation of the runtime of this approach assume that two time series T and T' with $|T| = k$ and $|T'| = n$ are to be aligned by the approach. Without loss of generality let $k \geq n$. The length of a predicted sequence is $n - 1$ in the worst case. The number of possible sequences that can be predicted is given by the size of the search space $|S'|$. We approximate $|S'|$ by $O(k \cdot \kappa) = O(k)$ for a suitable constant $\kappa \in \mathbb{N}$. The computation time for filling the matrix is $O(k \cdot n) = O(k^2)$ since every entry of the matrix is considered exactly once. The predicted sequences are calculated as well. This takes time $O(\sum_{i=1}^{k} i) = O(k^2)$ in the worst case, when every possible sequence is predicted. The overall time for the computation of one semiglobal alignment is therefore $O(k \cdot n + k^2) = O(k^2)$. In the case that k is the maximum length of any time series in the rule base, the overall running time of our algorithm is $O((k^2)|S'|) = O(k^3)$, since the observed time series is alignment with every typical context time series in the rule base S'

6.2.6 AR, MA and ARMA models

Despite recent developments with nonlinear models, some of the most common stochastic models in time series prediction are parametric linear models as autoregressive (AR), moving average (MA) or autoregressive moving average (ARMA) processes [147]. Examples for application scenarios for these methods are financial time series prediction scenarios and wind power prediction.

Assume a stochastic process $\pi(t)$ that generates outputs $\chi(t)$ at each point t in time. The random values $\chi(t)$ can be univariate or multivariate and can take discrete or continuous values and time can also be either discrete or continuous.

We are now interested in finding the parameters $\Theta = \{\theta_1, \ldots, \theta_n\}$ that describe the stochastic mechanism, for example, by maximising the likelihood that a set of values, $\{\chi(t_1), \chi(t_2), \ldots, \chi(t_k)\}$ were actually generated by that mechanism [117].

Forecasting or prediction is accomplished by calculating the conditional probability density $P(\chi(t)|\{\Theta, \{\chi(t-1), \ldots, \chi(t-m)\}\})$.

For moving average (MA) processes, let $Z(t)$ be some fixed zero-mean, unit-variance random process. $\chi(t)$ is a $MA(k)$ process or moving average process of order k, if $\chi(t) = \sum_{\tau=0}^{k} \beta_\tau Z(t-\tau)$, where the β_τ are constants. Moving average processes are utilised to describe stochastic processes that

have a finite, short-term linear memory [148, 149, 150]

In Autoregressive (AR) processes, the values at time t depend linearly on previous values. $\chi(t)$ is an $AR(k)$ process, or autoregressive process of order k, if $\sum_{\nu=0}^{k} \alpha_\nu \chi(t-\nu) = Z(t)$, where α_ν are constants. Autoregressive processes are used to capture exponential traces [148, 149, 150].

ARMA processes are a combination of AR and MA processes. An $ARMA(p,q)$ process is a stochastic process $\chi(t)$ in which $\sum_{\nu=0}^{p} \alpha_\nu \chi(t-\nu) = \sum_{\tau=0}^{q} \beta_\tau Z(t-\tau)$, where $\{\alpha_\nu, \beta_\tau\}$ are constants [148, 149].

Context prediction with ARMA methods

Since an ARMA process is already designed to approximate the development of a numeric time series in time, the only requirement for ARMA to be applicable to context prediction tasks is that the context types of all contexts in the observed time series are numerical.

Discussion

ARMA methods provide a powerful tool to approximate stochastic processes. The author of [6] also showed in his studies that ARMA processes are able to achieve excellent results in context prediction tasks.

The method is applicable to one-dimensional, as well as multi-dimensional, data sets alike. The computational complexity of the method is low and can be estimated as $O(k \log(k))$ [151].

No prior pre-processing or separate learning tool is required.

It is, however, only applicable to contexts of numeric context data type. In context prediction scenarios the method is hence not applicable to many problem domains.

6.2.7 Kalman filters

The Kalman filter is a set of recursive mathematical equations and provides an optimal way to estimate the state of a system from possibly erroneous observations [10].

Consider a one-dimensional system in which the state is represented by a vector $\overrightarrow{x_i} \in \mathbb{R}$ that is controlled by the equation

$$\overrightarrow{x_{t+1}} = \overrightarrow{x_t} + V_t, t = 1, 2, \ldots \tag{6.5}$$

The state of the system at time $t+1$ depends on the state $\overrightarrow{x_t}$ of the system at time t plus a random noise term V_t. An observation $\overrightarrow{y_t}$ of the system is represented by

$$\overrightarrow{y_t} = \overrightarrow{x_t} + W_t, t = 1, 2, \ldots \tag{6.6}$$

where the observation depends on the state of the system plus a random noise term W_t that describes the measurement inaccuracies. We are interested in how to determine the best approximation of $\overrightarrow{x_t}$, given $\overrightarrow{y_t}$. When we assume that the process noise V_t is a white Gaussian noise with covariance Q_t and that the measurement noise W_t is a white Gaussian noise with covariance R_t and further assume that W_t and R_t are uncorrelated, the prediction provided by the Kalman filter is optimal regarding the measurement error [107]. Note however, that these assumptions seldom hold. The Kalman filter is defined as

$$\overrightarrow{x_{t+1}} = \overrightarrow{x_t} + \frac{\Omega_t}{\Omega_t + R_t} \cdot (\overrightarrow{y_t} - \overrightarrow{x_t}), \tag{6.7}$$

$$\Omega_{t+1} = \Omega_t + Q_t - \frac{\Omega_t^2}{\Omega_t + R_t} \tag{6.8}$$

with $\Omega_0 = E[(y_0 - x_0)^2]$. Given an observation $\overrightarrow{y_t}$ at time t, together with a prediction $\overrightarrow{x_t}$, the approximation of the system state is equal to the previous state plus a term that is proportional to the distance between the last observation and the prediction. Intuitively, the higher the measurement noise R_t, the lower the impact of the observation in computing the next estimate. Vice versa, the higher the process noise Q_t, the higher the importance assigned to the latest observation. The Kalman equations thus both project the current state forward in time and incorporate new measurements in order to improve the estimate [107].

Context prediction with Kalman filters

Similar to the ARMA prediction approach, the Kalman filter is a stochastic method designed for forecasting numerical time series. Hence, for context elements of numeric context data type, it can be applied naturally. Examples for applications of the Kalman filter technique to Context-aware scenarios are [152, 153, 154].

Discussion

The Kalman filter computes a prediction based on an arbitrary long history of observed contexts. The computational load of the kalman filter method can be estimated as $O(3nm^3)$ [155].

Due to the high dynamic of context in pervasive settings, it is not feasible to train a Kalman filter to predict the actual values of context elements. It is possible, however, to predict the discrepancy between the correct context value d_t that will actually occur at some point t in time and the predicted value p_t for this time. Kalman filters are not applicable to context elements of non-numeric context data type.

We therefore believe that Kalman filters are not well-suited in context prediction scenarios.

6.2.8 Summary

From the prediction methods discussed, the alignment method turns out to be the most variable one as depicted in table 6.2.

In the table, several characteristics are depicted for various prediction methods. From these characteristics, the most important for context prediction are applicability to non-numeric contexts as well as to numeric contexts, the learning ability and the applicability to multi-dimensional and multi-type discrete time series.

Considering these four requirements, the Markov and the alignment approaches qualify as the best suited context prediction approaches.

The SOM is very similar to the Markov approach, only it is not applicable to non-numeric contexts. Support vector machines are not well-suited to context prediction tasks, since they are not applicable to non-numeric context types and also to discrete data sets that we consider as typical in context prediction scenarios. The Kalman filter and the ARMA method have quite similar features since they both represent statistical methods. Although the non-applicability to non-numerical data sets is a most serious drawback, respectable results have been obtained for the ARMA method in [6].

We consequently also consider an ARMA approach in the following discussion. The three prediction methods ARMA, Alignment and Markov which we consider as best suited for context prediction tasks, are therefore studied in the following sections in distinct scenarios.

	SOM	Markov	SVM	ARMA	Kalman	Align	State
Count of typical patterns	fixed[11]	variable	fixed[11]	0[12]	0[12]	variable	1
Numeric context types	yes	yes[13]	yes	yes	yes	yes	no
Non-numeric context types	no	yes	no	no	no	yes[14]	yes
Complexity[15]	$O\left(k^3\right)$	$O(k\lvert C\rvert^2)$	$O(k)$	$O(k\log(k))$	$O(k^4)$	$O(k^3)$	$O(k\lvert C\rvert^2)$
Learning ability	yes[16]	yes	yes	no [17]	no[17]	yes	no
Approximate pattern matching	yes	no	yes	no[12]	yes	yes	no
Multi-dim. TS	yes	yes	yes	yes	yes	yes	yes
Discrete data	yes	yes	no	yes	yes	yes	yes
Variable length typical patterns	yes	no	no	no[12]	no[12]	yes	no
Multi-type TS	no	yes	no	no	no	yes	no
Continuous data	no	yes	yes	yes	yes	no	no
Pre-processing	yes	no	yes	no	no	yes	no
Prediction of durations	yes	no[18]	yes	yes	yes	yes	no
Continuous time	no	yes	no	yes	yes	no	no

Table 6.2: Properties of context prediction approaches.

[11] Variable if pre-processing process is constantly iterated with the observation of new contexts.
[12] No such measure for the ARMA approach and for Kalman filters.
[13] Clustering of values in advance required.
[14] Depending on similarity metric.
[15] Runtime for a context prediction horizon of $n = O(k)$ context elements.
[16] Repetition of model creation phase required.
[17] Method adapts to last observed context time series.
[18] Possible when time is also considered a context source.

Observe that the discussion is restricted to those prediction methods that we deem most suited for context prediction tasks. Undoubtedly, further statistical methods or variations of the named methods exist that can also be applied to obtain information on the continuation of a time series. However, ARMA, MA, AR and Kalman filters are chosen as representatives for the class of statistical methods.

In addition, neural networks and evolutionary algorithms can be applied to the task of time series prediction. Neural network approaches are, for example, applied in [85]. The results obtained challenge the ARMA algorithm on numerical data for short prediction horizons. However, since neural networks have no repository of typical contexts, the general operation is similar to the ARMA method. Both approaches transform all information contained in the most recently observed context time series to a time series that describes the evolution in the future. Since ARMA is specialised for this task, we expect it to outperform the neural network approach. Evolutionary algorithms on the other hand, are in our opinion computationally too expensive to be applicable to mobile ubiquitous scenarios.

A compromise is taken in [85] where particle swarm optimisation (PSO) methods are utilised. Although computationally less expensive, PSO-techniques are known to quickly collapse to local optima in the search space. Consequently, we deem this approach not well-suited for context prediction tasks.

Additionally, other search algorithms like, for example, simulated annealing, might be adapted to context prediction tasks. However, our choice of algorithms represents the most commonly applied and most straightforward approaches. Since we are not aware of any attempt to apply these approaches to context prediction tasks, these methods are not considered. However, further research is required in order to be able to estimate the performance of these rather classical methods for context prediction tasks.

Further methods we did not consider in detail are the IPAM algorithm presented in [86], as well as graph based models [99, 87]. For the IPAM algorithm, it has already been shown that the accuracy is very low. For the path learning algorithms, they are very similar to Markov prediction algorithms so that we did take the Markov approach as a representative of this prediction method.

6.3 Implementation

The following sections present implementation details for the algorithms utilised in the simulations. Implementation related descriptions are provided for the alignment and the Markov prediction approaches as they have been implemented by us in the course of this work. For the ARMA prediction approach we utilised the JMSL Java library[19].

6.3.1 Alignment prediction approach

For the alignment prediction approach we combine alignment techniques as described in [106] with prediction capabilities. The alignment problem is solved by the Needleman and Wunsch algorithm [127]. The prediction task follows the procedure described in section 6.2.5. In the following sections we describe the representation of context elements in our implementation, the distance metric and the similarity threshold.

Time series elements

We represent time series as concatenated vectors in an n-dimensional vector space. A single entry in a time series contains information from various context sources. Before being able to compare two entries, we normalise all data to $[0, 1]$. Each context element is unambiguously represented by a point in an n–dimensional coordinate system where n is the number of context sources involved. The values of each single context source are uniquely represented by one axis of the coordinate system. A short illustrative example for numeric context types is given in figure 6.4.

Note that the representation of the time series element as a point in the n-dimensional vectorspace is only suitable with numerical contexts. However, the approach can be applied to arbitrary context types. In this case, the mapping onto a coordinate system might be not trivial but is generally feasible in all cases.

Distance metric

In order to rate the similarity between two context time series T_1 and T_2, a distance metric $d : T \times T \rightarrow \mathbb{R}$ is required. The score of the optimal semiglobal alignment is calculated as the sum of the distances between all

[19] http://www.vni.com/products/imsl/jmsl/jmsl.html

	$s_{\max}$	$s_{\min}$	$S(i)$	$S(i) \cdot \frac{1}{s_{\max} - s_{\min}}$
Context source A	10	0	5	0.5
Context source B	5	−3	3	0.375
Context source C	7	2	1	0.2

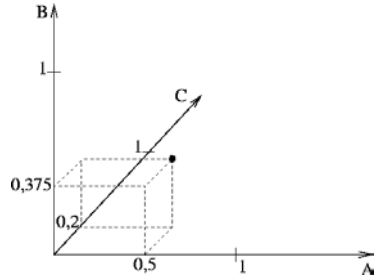


Figure 6.4: Values for three context sources are mapped into a coordinate system.

context time series elements ξ_i with the same index ($L_n : \xi \times \xi \rightarrow \mathbb{R}$). In our case, we obtain therefore $d(T_1, T_2) = \sum_{i=1}^{l} L_n(\xi_i, \xi_i')$ with l being the length of the local alignment.

To compare two entries in possibly different time series we compare the corresponding points in the n-dimensional hyperspace. The correlation between any two points ξ_i and ξ_j in this hyperspace is given by the Euclidian distance. For non-numerical contexts, an alternative distance metric has to be applied.

The maximum distance according to the distance metric with respect to the dimensionality n of the hyperspace is $\sqrt[n]{n}$. Therefore, we define two points in the coordinate system to be similar to each other if the distance is less than $\frac{1}{r}\sqrt[n]{n}$ for proper r. We define the distance between any point in the coordinate system to the symbol $\{-\}$ to be $\frac{1}{r}\sqrt[n]{n}$ because of our assumption that a gap symbol is considered a missing symbol in the sequence instead of a completely different symbol. If we search for optimal semiglobal alignments of a minimum length of l, we require the optimal semiglobal alignment to have a maximum total cost of $K = \frac{d}{r}\sqrt[n]{n}$.

Similarity threshold

In order to enable an adaptive operation of the alignment prediction method, new context patterns might be appended to the rule base at runtime. Every time an observed context time series is not similar to any of the typical time series in the rule base, it is itself considered a new typical time series and added to the rule base.

We define a similarity threshold τ. A time series T_1, whose similarity value compared to a time series T_2 is below τ is considered similar to T_2.

6.3.2 Markov prediction approach

The Markov prediction implementation receives context elements upon a context change and predicts future context changes one or more timestamps ahead. For the modelling of the Markov model, in particular the transition probabilities between contexts, we maintain a matrix of context transition probabilities. The rows and columns in this matrix represent observed and predicted contexts respectively. The prediction of the most probable next context is then obtained by finding the column of the entry with the highest transition probability in the row that corresponds to the current context. Given all information available at the current time interval, a prediction of contexts farther away in the time horizon is obtained by iteratively multiplying the matrix with itself. After $n-1$ multiplications, the resulting matrix represents the n-step ahead prediction.

Let $k \in \mathbb{N}$ be the length of the context history. The Markov implementation represents an order-k Markov model. The corresponding matrix of context transitions is consequently not only composed of rows that correspond to one distinct context element, but it is rather composed of rows that correspond to context time series of length k.

A Markov prediction approach naturally does only support the prediction of context values. A prediction of time stamps that are associated with these values is not obtained.

Matrix of context transitions

The Markov implementation maintains a matrix of context sequences. Every time a context change occurs, the matrix is updated and the probabilities are recalculated. The matrix is stored in a hash map in which

the sequence of contexts is utilised as the key and information on that sequence as the value of the hash map.

The information associated with one context sequence contains data on how many times this sequence appeared in series of received contexts and all its successor contexts as well as the number of their occurrences. According to these statistics the probabilities for all possible successors are calculated.

Sequences are implemented as linked lists of elements representing contexts and their values. Contexts are distinguished between by a unique ID. This unique ID is identical for all context elements of the same context type.

Due to this concept we are able to represent multi-dimensional and multi-type context time series in the transition matrix.

Prediction procedure

The Markov predictor computes contexts of one distinct time stamp in a single step. Predictions for several steps ahead require several distinct predictions of iteratively increasing context horizon. After every matrix multiplication the probabilities of single contexts need to be extracted from the transition matrix. The matrix is afterwards again multiplied with the one-step ahead transition matrix in order to obtain the most probable contexts one step further ahead in time. This procedure is iterated until the desired length of the context prediction horizon is reached.

In the case that several columns in the row that is utilised for prediction purposes share the maximum transition probability, all corresponding contexts are considered for prediction.

6.4 Simulations

In the following section we present results obtained for the ARMA, the Markov and the alignment approaches in simulations on real data. The simulation scenarios are chosen from different problem domains in order to allow for a domain-independent discussion of the simulation results obtained by the algorithms.

Since the ARMA approach is exclusively applicable on context elements of numerical context data type, the simulation scenarios cover only numerical context data.

For the ARMA algorithm we expect a good approximation of the general trend of the observed context time series, since this constitutes the strength of the method. For the Markov approach we expect best results for hardly fluctuating context time series values, since low fluctuation corresponds to a strong typical context pattern that can be predicted with high probability. With highly fluctuating time series, a high number of low-probability typical patterns can be associated.

Finally, from the alignment approach we expect a good performance when the observed context time series is a random concatenation of typical context patterns that have low similarity to each other, since this concatenation can directly be found by concatenating the typical time series stored in the rule base of the algorithm.

6.4.1 Windpower prediction

Wind is one of the inexhaustible energy sources humans have access to. In order to utilise this power, wind farms are built that consist of a multitude of windturbines. The amount of power available from these parks is constantly fluctuating and heavily dependent on the wind power. Since the power supply system is hardly capable of dealing with these instantaneously fluctuating power curves, methods that predict the wind power are required to schedule the power expulsion of other power sources in order to balance the power level.

In this section, we compare the prediction accuracy achieved by the alignment prediction approach [8] on a realistic data set to the accuracy achieved by a Markov prediction method and implementations of ARMA prediction algorithms.

Simulation scenario

The data set utilised for the simulation contains wind power samples from a wind farm in Germany. The data samples have been recorded hourly from February 2004 to April 2005. We apply two context prediction algorithms to the data set. These are a Markov approach of order 6 and the alignment prediction method. Both algorithms have their context history size restricted to 6 hours. We complete various simulation runs with differing context prediction horizon of the algorithms. Prediction results for prediction horizons ranging from 1 hour to 20 hours are computed.

In addition, we utilise an ARMA approach with an unconstrained context history as upper bound. Since the ARMA algorithm has an unrestricted context history, the knowledge about the prediction process is much higher for this method. For comparison, we apply a modified version of the ARMA algorithm (ARMA-100). For ARMA-100, the context history size is restricted to 100 hours.

We utilised $\frac{3}{4}$ of the data samples as training data for the Markov and alignment algorithms and the remaining part for the simulation. The ARMA approach starts the first prediction only after $\frac{3}{4}$ of the data has been observed. The time series describing the wind power values contains real valued samples that range from 0 to 23.

To give an estimation on the accuracy of the prediction, we calculate the RMSE and BIAS for every algorithm as

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} \frac{(p_i - d_i)^2}{23}}{n}} \tag{6.9}$$

$$BIAS = \frac{\sum_{i=1}^{n} |p_i - d_i|}{23 \cdot n} \; . \tag{6.10}$$

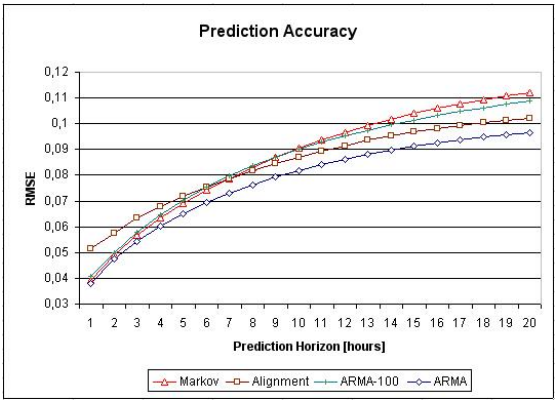
In these formulae, p_i depicts the predicted value at time i while d_i is the value that actually occurs at time i. The value of 23 is utilised for normalisation since 23 is the maximum value in the data set.

Simulation results

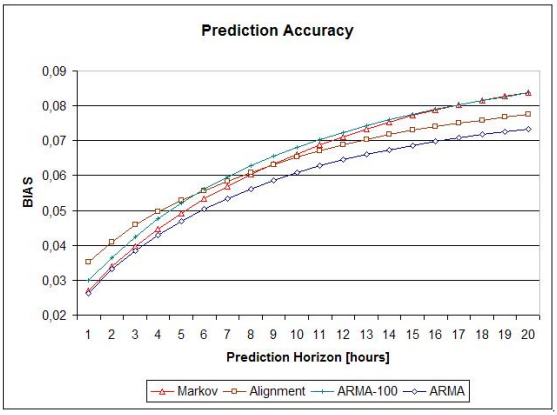
The results of the simulation regarding the RMSE and BIAS values are depicted in figure 6.5.

From the figure, we observe that both ARMA and Markov methods are nearly identical for short prediction horizons, while the alignment prediction approach performs worse. However, with prediction horizons that exceed the context history size (6 hours and more), the alignment prediction algorithm outperforms both the Markov and the ARMA-100 implementations. Still, it does not reach the same accuracy as the unrestricted ARMA approach.

When comparing the ARMA with the alignment approach, we have to consider the different resources both acquire. While the ARMA approach has access to the complete context history, the alignment approach utilises,



(a) Prediction accuracy measured by the RMSE metric.



(b) Prediction accuracy measured by the BIAS metric.

Figure 6.5: Prediction accuracies for the ARMA, Alignment and Markov prediction algorithms.

apart from the information extracted from the training data, only a context history of six hours. However, it extracts and stores typical context patterns from the observed context sequence.

As it can be observed from the figure, with a context history restricted to 100 hours, the ARMA-100 approach is already worse than the alignment approach for higher prediction horizons. If the context history size of the ARMA approach is further reduced (eg to 50 hours and less), the simulation has shown that the ARMA approach is by far worse than the alignment algorithm for all prediction horizons.

Results for the BIAS metric are similar. The major difference to the RMSE results is that the Markov algorithm performs marginally worse and that the alignment prediction algorithm outperforms the Markov method already at a prediction horizon of 5 hours. Furthermore, for higher prediction horizons the accuracy of the Markov approach becomes nearly identical to the ARMA-100 algorithm.

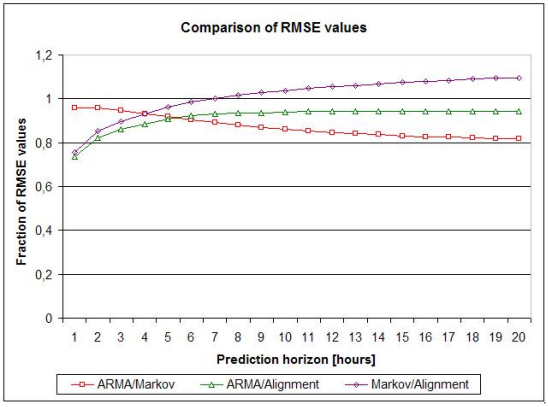
A direct comparison of these results for the prediction algorithms is given in figure 6.6. In these figures, the fraction of the BIAS and RMSE results of two algorithms is depicted. For pairs of algorithms A_1 and A_2 the figure depicts the fraction $\frac{RMSE(A_1)}{RMSE(A_2)}$. The figures again show that the unrestricted ARMA approach outperforms the alignment and the Markov-algorithms, while the Markov then is only second best to the alignment approach.

Summary

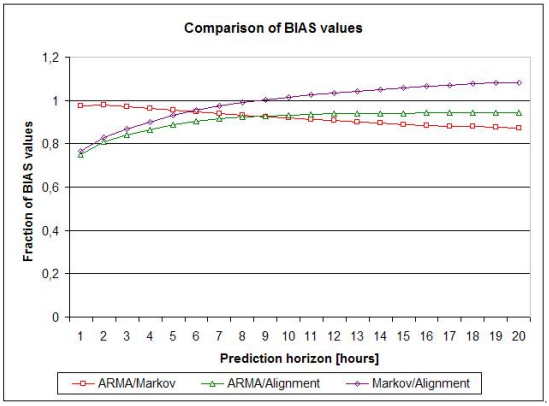
The simulation shows that out of the choice of context prediction algorithms, the ARMA approach is best suited for context prediction that is applied to a data set of wind power values. This result follows our expectation on the performance of the prediction algorithms. In a scenario where the evolution of a numeric data set is dominated by trends and periodic patterns, the ARMA approach is most well-suited. However, also the alignment approach scores respectable results on this data set.

We have observed that the alignment prediction approach performs well on the data set for high prediction horizons, whereas for low prediction horizons the Markov and ARMA-100 approaches should be preferred.

These results further show that the alignment prediction approach is well-suited to extract plenty of information even from short context histories.



(a) Comparison of prediction accuracies with the RMSE metric.



(b) Comparison of prediction accuracies with the BIAS metric.

Figure 6.6: Comparison of prediction accuracies for the ARMA, Alignment and Markov prediction algorithms.

This property makes the alignment prediction approach especially well-suited for ubiquitous computing environments. Since the context evolution process might be described by a series of short characteristic context patterns, it is essential that the prediction algorithm is capable of extracting a maximal amount of information from these patterns.

6.4.2 Location prediction

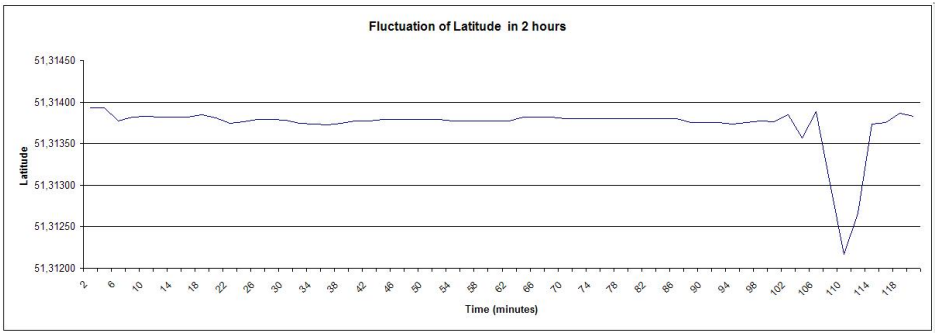
In order to compare the context prediction algorithms also in an environment that is more common for context-aware computing than the prior wind-power prediction example, we consider a scenario in which the location trajectory of a mobile user is to be predicted. The overall setting is identical to the one described in section 5.1.3.

For this scenario we again focus on the overall performance of the prediction algorithms, as well as on the impact of the prediction horizon on the context prediction accuracy. For this scenario, also the sampling interval in which samples are taken might influence the prediction accuracy. As we have discussed in section 5.1.3, with a small sampling interval, the idle periods might dominate the the context changes in the sampled data. In this case, the extrapolation of the current context value into the future becomes more beneficial for the prediction algorithm, than the actual prediction of important context changes.

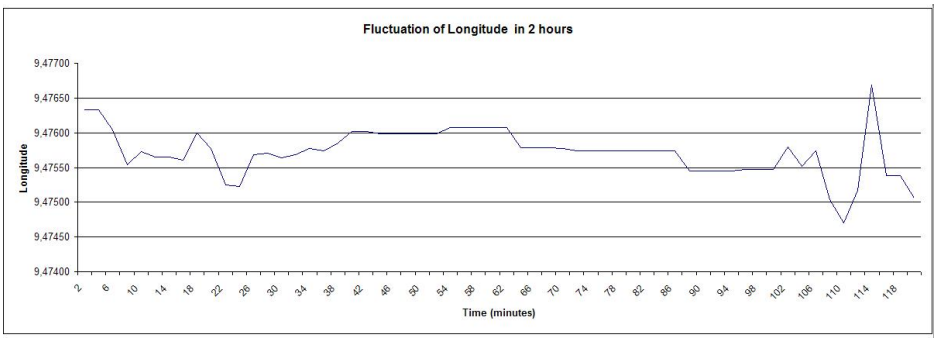
Since the optimal data-sampling frequency for this scenario is not known and furthermore dependent on the typical user behaviour patterns, we consider several data sampling intervals for all prediction algorithms.

In order to find a sampling interval that is best suited to this environment, we analyse the sampled data for the frequency of context changes. A context change for the GPS data is basically a change in the measured GPS position. Due to errors in the GPS measurements, the sampled GPS data is fluctuating even though the user position did not change. Figure 6.7 depicts measurements of the sampled GPS data. As you can observe from the figure, the fluctuation is easily in the order of 30 to 50 meters in a few minutes, although the general user position is more or less static. We therefore define a context change as a GPS measurement that differs to a substantial grade to the last context change.

In order to find a distance between two GPS measurements that are most suitable to define a context change in our case, we analyse the number of context changes in the sampled data for several distances (cf. table 6.3).



(a) Fluctuation of measured latitude.



(b) Fluctuation of measured longitude.

Figure 6.7: Fluctuation of location measurements.

Distance	average time till context changes
5 meters	6.866 minutes
10 meters	8,9846 minutes
20 meters	13.3859 minutes
30 meters	17.175 minutes
50 meters	24.9622 minutes
100 meters	43.0246 minutes

Table 6.3: Average times between two context changes

From these values we observe that a small distance threshold is likely to produce lots of context changes. However, since the mobile user did not change her position every seven minutes in the three weeks the experiment lasted, we assume that many of these context changes are due to measurement inaccuracies. It is more reasonable to assume that relevant context changes appear only every 20 to 40 minutes on average, since during the night hours we do not even expect a context change at all.

If the context durations are heavily fluctuating, the context algorithm might either miss out contexts since the context sampling interval is too large, or might on the other hand miss relevant context changes since the context durations are too long, and consequently the relevant context changes are too few compared to irrelevant context changes to make out a typical context pattern. A too frequent measurement of GPS samples might mislead the prediction algorithm in a way that it predicts the random measurement inaccuracies, rather than relevant user context changes. Therefore, we choose sampling intervals of 12 and 20 minutes respectively, in order to recognise the important context changes that we expect to happen every 20 to 40 minutes on average. This sampling frequency is assumed to be sufficient to describe the context time series of the user in a granularity that suffices to model the relevant context fluctuation, without sampling too much unimportant process noise information and without missing relevant context changes.

Simulation results

The prediction accuracy of the prediction algorithms is again measured by the RMSE and BIAS metrics (cf. section 3.2.4). Figure 6.8 depicts the fraction of prediction accuracies for various prediction horizons. In the figure, pairs of algorithms A_1 and A_2 are compared for their prediction accuracies. The figure depicts $\frac{RMSE(A_1)}{RMSE(A_2)}$ for sampling intervals of 12 and 20 minutes and for various prediction horizons.

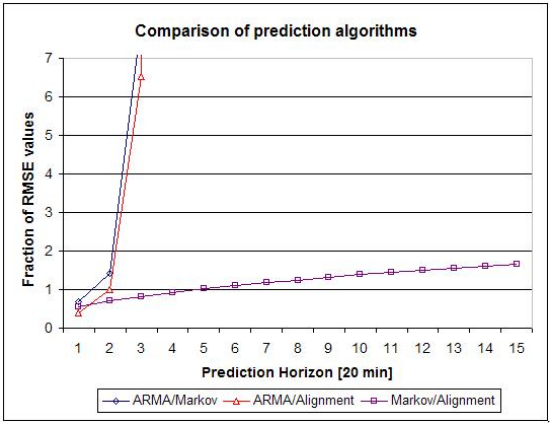
With a sampling interval of 20 minutes, obviously the ARMA algorithm is not competitive compared to the other prediction algorithms. For the comparison of the alignment and the Markov approaches, we again observe that the Markov-algorithm is to be preferred for short prediction horizons of up to 80 minutes, while the alignment algorithm performs better for higher prediction horizons.

For a sampling interval of 12 minutes the ARMA approach is again competitive. We observe that the fraction $\frac{RMSE(ARMA)}{RMSE(Markov)}$ is approximately 0.6 for all prediction horizons considered. Again the Markov-approach outperforms the alignment algorithm for short prediction horizons, while it is complementary with the increase of the prediction horizon. However, compared to the alignment-approach, the Markov-algorithm performs slightly better than for the sampling interval of 20 minutes.

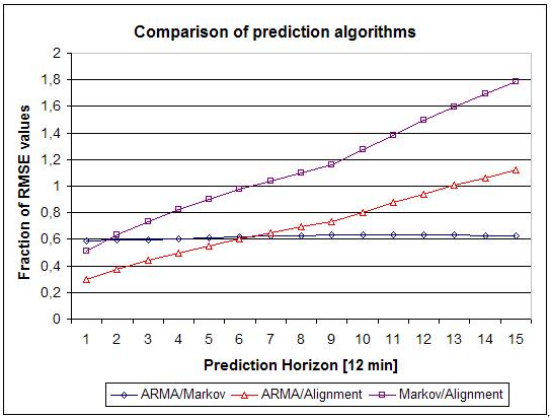
Compared to the ARMA approach, the interpretation for the alignment algorithm is similar but shifted to longer horizons in favour of the ARMA approach. Only at prediction horizons of 2.5 hours and more does the aliment approach outperform the ARMA algorithm. For the BIAS values the results are similar (cf. figure 6.9).

Summary

These results show that the alignment approach is especially well-suited for context sequences that inherits typical patterns and in scenarios where a high accuracy for longer prediction horizons is required. For shorter prediction horizons, the ARMA approach may be preferred. The Markov prediction algorithm, however, is inferior to at least one of the other prediction approaches for all configurations considered. Generally, the ARMA algorithms scores the better results of these two, however, for high prediction horizons the alignment prediction approach computes the more accurate results. Additionally and most seriously, for high sampling intervals we observe that the ARMA approach is not competitive.

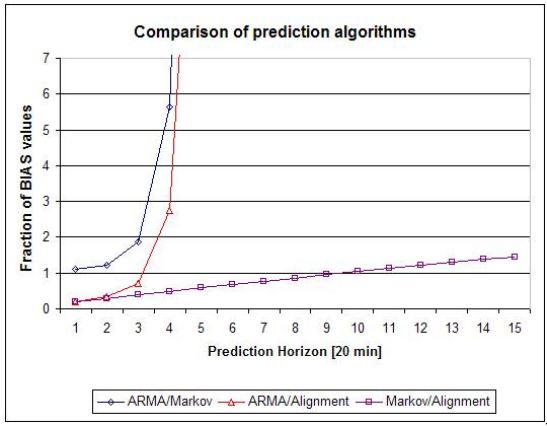


(a) Sampling interval: 20 minutes.

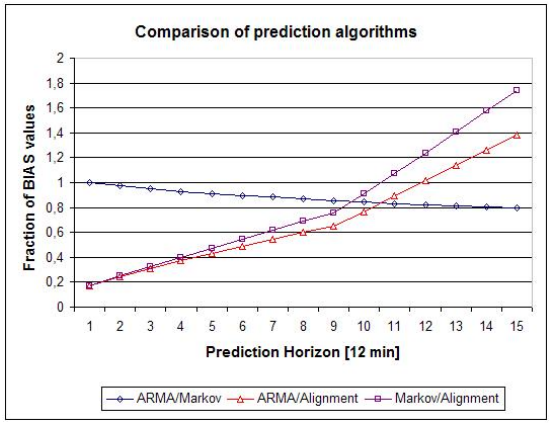


(b) Sampling interval: 20 minutes.

Figure 6.8: Comparison of the ARMA, Markov and alignment algorithms for the RMSE metric with various sampling intervals and prediction horizons.



(a) Sampling interval: 20 minutes.



(b) Sampling interval: 12 minutes.

Figure 6.9: Comparison of the ARMA, Markov and alignment algorithms for the BIAS metric with various sampling intervals and prediction horizons.

With the knowledge of the average prediction accuracies for a given scenario, a hybrid approach as suggested in [7] might therefore be beneficial. The general idea of this approach is to utilise the prediction result of that algorithm that has the lowest probability of error in a given situation.

6.5 Summary

In this section we have discussed various requirements for context prediction algorithms. Various context prediction approaches have been discussed regarding their capability to fulfil the specified requirements. We identified three algorithms, namely the alignment prediction approach, the Markov prediction method and the ARMA algorithm, to be most suited for context prediction in ubiquitous computing environments.

For implementations of these algorithms we have conducted simulations for two distinct simulation scenarios. As an excerpt of both simulations we observe that for low prediction horizons, the ARMA approach was the dominating prediction method, while for longer prediction horizons the alignment prediction approach became more competitive and outperformed the ARMA approach in the location prediction case. While the good results of the ARMA approach on these numerical data sets could be expected, especially the alignment prediction performance is remarkable. We have especially observed that over time the alignment approach best adapts to a given scenario.

Furthermore, the alignment approach is capable of achieving competitive results already with a limited knowledge of the observed context history.

7 Conclusion

> *I may not have gone where I intended to go, but I think I have ended up where I intended to be.*
>
> (Douglas Adams, Mostly harmless [156])

In the preceding chapters we have discussed issues related to context prediction in ubiquitous computing domains. In particular, the three main contributions of this thesis are an algorithm for context prediction that utilises alignment techniques, an architecture for context prediction on arbitrary context abstraction level as well as a comprehensive study of the impact of the context abstraction level on the context prediction accuracy.

The following sections summarise and detail the contribution of the thesis. In section 7.1, a brief overview on context prediction domains is provided and current challenges in context prediction domains are identified. We recapitulate solutions to these challenges that are provided by this thesis in sections 7.1.1 through 7.1.3. Section 7.2 gives an outlook on expected developments in context prediction domains.

7.1 Contribution

Recent advances in context prediction domains have provided architectures for context prediction [6, 4] as well as novel algorithms [86, 87, 157] and simulation studies [120, 100]. However, context prediction is currently only seldom utilised for context aware computing applications. We identified three reasons that hinder the broad utilisation of context prediction mechanisms in context-aware applications. First of all, the architectures for context prediction, that have been proposed so far, are restricted in the

abstraction level of the input data. An application to arbitrary context prediction approaches is therefore not possible.

Furthermore, the success of context prediction approaches depends on the accuracy of predictions. Regarding the context prediction accuracy, not all mechanisms and interdependencies are yet understood.

Finally, algorithms utilised for context prediction tasks are often lent from related prediction domains. Consequently, these approaches are possibly not domain specific and might be suboptimal for a given prediction scenario.

This thesis presents solutions to these challenges. In particular, the following issues have been addressed.

Impact of the context abstraction level on the prediction accuracy

- Distinction between the two context prediction schemes 'low-level context prediction' and 'high-level context prediction'
- Development of an analytic model to estimate the accuracy of context prediction schemes
- Discussion of the impact of the context abstraction level on the context prediction accuracy
- Discussion of the impact of the context abstraction level on the learning quality of context prediction architectures
- Discussion of the impact of the search space dimension on the context prediction accuracy

Algorithms for context prediction

- Design and Implementation of the alignment prediction approach
- Comparison of context prediction algorithms in two context prediction scenarios

Standards and definitions

- Definition of context sources
- Definition of context abstraction levels with regard to the amount of pre-processing applied
- Definition of context time series
- Definition of the context prediction task

- Refinement of a definition of context types
- Refinement of a definition of context data types
- Design and implementation of a lean and flexible architecture for context prediction.

These contributions are detailed in the following sections.

7.1.1 Impact of the context abstraction level on context prediction accuracy

We distinguished between high-level and low-level context prediction schemes in chapter 3 and considered impacts that originate from distinct context abstraction levels of the input data of context prediction methods. A special focus was laid on the processing of contexts, the long-term accuracy and the search space dimension of various context abstraction levels.

Impacts of the context abstraction level on the context prediction accuracy have been studied analytically and on real and synthetic data sets. We have developed a set of formulae that describe the context prediction accuracy of low-level and high-level context prediction schemes with respect to environment specific parameters. In particular, the context history length, the prediction horizon, the count of low-level and high-level contexts and context values at one time interval, as well as error probabilities for various context processing modules, have been considered.

In the analysis we observed a benefit of the low-level context prediction scheme above the high-level context prediction scheme for most parameters considered. High-level context prediction schemes are especially disadvantaged when the number of high-level contexts in every time interval is high, as well as for scenarios in which the information necessary for prediction purposes is spread over many time intervals in the context history. In simulations we also observed that the high-level context prediction scheme is disadvantaged when long prediction horizons are required. Both context prediction schemes are equally vulnerable to context acquisition errors. Regarding the impact of the dimension of the input time series on the size of the context history, an increase in dimension fosters the low-level context prediction scheme while this predominance diminishes with increasing context history size.

We further demonstrated that the learning method of a high-level context prediction scheme is impaired when compared to the learning method

of a low-level context prediction scheme. Due to the higher probability of error in the input sequence, the learning method of a high-level context prediction algorithm might adapt to erroneous context patterns. Such an adaptation to erroneous sequences impacts the instantaneous as well as the long-term prediction accuracy.

Additionally, the size of the search space impacts the context prediction accuracy. Context time series of higher context abstraction level likely correlate to a decreased size of the search space. While this constitutes a benefit in highly error prone scenarios, since the time series are more robust to errors in the input sequence, the information content of time series is reduced. This leads to a less detailed description of context sequences and more likely to ambiguous prediction alternatives. The prediction accuracy decreases especially in complex scenarios for the high-level context prediction scheme.

We also considered the memory consumption of both context prediction schemes. Due to the lower information content of the prediction search space, the high-level context prediction scheme has a lower memory consumption than the low-level context prediction scheme. However, this benefit comes at the cost of accuracy.

7.1.2 Algorithms for context prediction tasks

Algorithms utilised for context prediction tasks are, on the one hand, algorithms prominent in related prediction domains and on the other hand, novel, domain specific prediction approaches. While the former approach has the benefit that the design and development are less intensive, it also holds the danger of loosing accuracy by the adaptation of an algorithm that is not well suited for context prediction domains.

In chapter 6 we identified requirements for context prediction algorithms in ubiquitous computing environments. Most importantly, these are the prediction accuracy, as well as adaptability and resource consumption. Under consideration of these requirements the alignment prediction approach was developed. This algorithm constitutes a novel approach to the task of context prediction. The algorithm is especially well suited in scenarios where repeatedly occurring short context patterns characterise the observed time series.

The alignment prediction approach has been compared to common prediction approaches regarding the requirements identified. It was further

compared in simulations on two real data sets against an ARMA and a Markov prediction approach. The algorithm performed well in both simulations. Especially for a context history of restricted size, as well as for a high prediction horizon, the alignment prediction algorithm outperformed the competing prediction approaches.

7.1.3 Standards and definitions

In the course of this thesis, we have utilised concepts from related work as, for example, the notion of context or context awareness and the notion of features of contexts but have also developed new concepts or improved existing ones were necessary.

In chapter 2, the notion of a context source was introduced as a construct to encapsulate features of sensors. We also proposed a distinction of context abstraction levels on the basis of the amount of processing applied to the context elements. This distinction differs from recent descriptions of context abstraction levels that cluster contexts by human-intuitive notions. Our more technical distinction allows a formal discussion of issues related to the context abstraction level. Additionally, we distinguish contexts by their 'context type' as well as by the 'context data type'. Both concepts are revived from the literature but further detailed. We expanded the model of context data types to hierarchical contexts that have not been considered before. Our distinction of context types combines concepts proposed by Abowd, Dey, Brown, Davies, Smith, Steggles as well as by Mayrhofer, Radi and Ferscha. We further detailed the model and completed it by the aspects 'relative location', 'relative time', 'biological', 'mood' and 'equipment'.

In chapter 3, the definitions of 'context time series elements', dimensions of context time series, realistic context time series as well as the 'context history' have been introduced. These concepts are required for a formal discussion of the context prediction accuracy and have been utilised consistently throughout the thesis.

We also developed a formal definition of the context prediction task in the sense that it specifies the input data, the output data and the aim of the prediction task. In particular, context prediction is a search problem that requires context time series as input and output data. The prediction aim is to approximate a time series whose evolution is described by a probabilistic process. One major emphasis of this definition is the tight link of

learning to the context prediction task which was motivated by the realisation that UbiComp environments necessitate a learning capability. The proposed definition constitutes the first definition of context prediction in UbiComp environments.

In chapter 4 we developed general requirements for context prediction architectures. These requirements include the possibility of an adaptive operation, a distributed operation, as well as the applicability to context time series. Given these requirements, we designed and implemented a flexible and lean architecture for context prediction. It is flexible in the sense that its components may be distributed among computing devices in a network environment and in particular in the sense that it is applicable to contexts of arbitrary context abstraction level. This one-layered approach supports the identified requirements and follows a modular design. The feasibility of the architecture was demonstrated in application scenarios on real and synthetic data sets.

We have implemented three prediction modules for the context prediction architecture. These are an alignment approach, a Markov approach and an ARMA approach respectively.

The proposed architecture for context prediction widens the scope of application scenarios, context prediction approaches are applicable to. Especially the possibility to apply a context prediction mechanism at arbitrary context abstraction level is not covered by any other architecture for context prediction proposed in the literature.

7.2 Outlook

Although a sound level of theoretical and experimental knowledge concerning context prediction has been established, some interesting scientific questions remain. We briefly outline those questions we deem most relevant.

In our discussion on context prediction accuracies for high-level and low-level context prediction schemes we considered distinct values to represent sensor readings. However, since a measured value actually constitutes that most probable value in a probability distribution of neighbouring values, the consideration of probability distributions instead of distinct values for measurements might contribute to further increase the accuracy of the calculations.

We observed that the robustness of a prediction scheme increases with

lower search space dimension, whereas the capability to adequately describe context patterns decreases at the same time. A more decent understanding of this impact and especially an estimation of interrelations between these two opposing properties will foster the context prediction accuracy.

Furthermore, the concepts of remote prediction and prediction sharing constitute intriguing research issues. When predictions are computed by remote hosts, the prediction quality might improve since higher processing and memory capabilities allow for more ambiguous prediction algorithms. On the other hand, a connection loss then imposes a most serious thread for a continuous operation. However, a centralised operation enables also the implementation of concepts as prediction sharing or time series sharing. The benefits and possible risks of these approaches are not studied comprehensively yet.

Another most interesting approach is the redundant choice of context sources in a way that keeps the cost of an installation low while increasing the ability to detect and correct measurement errors at the same time (cf. section 5.1.2).

Furthermore, the joint processing of multi-type context time series might improve context prediction accuracies. As we have derived in chapter 5, the most suitable context abstraction level depends on properties of a given context. When contexts of distinct abstraction level are utilised concurrently, the most profitable abstraction level for each single context type might be chosen.

Given these chances and challenges in context prediction domains we believe that context prediction still constitutes intriguing open issues. Hopefully, we will see the results of fruitful research in the areas mentioned in the near future.

Index

List of Tables

List of Figures

Bibliography

[1] van der Duin, P., Kok, R.: Mind the gap - linking forecasting with decisionmaking. **4**(100) (2004) 185–194

[2] Want, R., Schilit, B., Adams, N., Gold, R., Petersen, K., Goldberg, D., Ellis, J., Weiser, M.: An overview of the parctab ubiquitous computing experiment. In: IEEE personal communications. Volume 2. (1995) 28–43

[3] Gellersen, H.W., Beigl, M., Krull, H.: The mediacup: Awareness technology embedded in an everyday object. In Gellersen, H.W., ed.: First International Symposium on Handheld and Ubiquitous Computing (HUC99), Lecture notes in computer science. Volume 1707., Springer (1999) 308–310

[4] Nurmi, P., Martin, M., Flanagan, J.A.: Enabling proactiveness through context prediction. In: CAPS 2005, Workshop on Context Awareness for Proactive Systems. (2005)

[5] Mayrhofer, R.: Context prediction based on context histories: Expected benefits, issues and current state-of-the-art. In Pronte, T., Beyers, B., Fitzpatrick, G., Harvel, L., eds.: Proceedings of the 1st international Workshop on exploiting context histories in smart environments (ECHISE) at the 3rd Int. Conference on Pervasive Computing. (2005)

[6] Mayrhofer, R.M.: An Architecture for Context Prediction. PhD thesis, Johannes Kepeler University of Linz, Altenbergstrasse 69, 4040 Linz, Austria (2004)

[7] Petzold, J.: Zustandsprädiktoren zur Kontextvorhersate in Ubiquitären Systemen (in German). PhD thesis, University of Augsburg (2005)

[8] Sigg, S., Haseloff, S., David, K.: Context prediction by alignment methods. In: Poster Proceedings of the fourth international conference on Mobile Systems, Applications, and Services (MobiSys). (2006)

[9] Garlan, D., Siewiorek, D., Smailagic, A., Steenkiste, P.: Project aura: Toward distraction-free pervasive computing. IEEE Pervasive computing **4** (2002) 22–31

[10] Capra, L., Musolesi, M.: Autonomic trust prediction for pervasive computing. In: Proceedings of IEEE Workshop on Trusted and Autonomic Computing Systems 2006 (TACS'06). (2006)

[11] Ferscha, A., Holzman, C., Leitner, M.: Interfaces everywhere – interacting with the pervasive computer (2006) Half-day tutorial, 10th ACM International Conference on Intelligent User Interfaces (IUI 2006), Sydney, Australia.

[12] Want, R., Hopper, A., Falcao, V., Gibbons, J.: The active badge location system. In: ACM Transactions on Information Systems. Volume 1. (1992) 91–102

[13] Weiser, M.: The computer for the 21st century. In: Scientific American. Volume 3. (1991) 66–75

[14] Dourish, P.: What we talk about when we talk about context. In: Personal and Ubiquitous Computing. Volume 8. (2004)

[15] Schilit, B.N., Theimer, M.M.: Disseminating active map information to mobile hosts. In: IEEE Network. Volume 5. (1994) 22–32

[16] Schilit, B.N., Adams, N., Want, R.: Context-aware computing applications. In: IEEE Workshop on Mobile Computing Systems and Applications. (1994)

[17] Brown, P.J., Bovey, J.D., Chen, X.: Context-aware applications: from the laboratory to the marketplace. In: IEEE personal communications. Volume 4. (1997) 58–64

[18] Pascoe, J.: The stick-e note architecture: Extending the interface beyond the user. In: Proceedings of the 2nd International Conference on Intelligent user Interfaces. (1997) 261–264

[19] Dey, A.K., Abowd, G.D., Wood, A.: Cyberdesk: A framework for providing self-integrating context-aware services. In: Knowledge-Based Systems. Volume 11. (1998) 3–13

[20] Schmidt, A., Beigl, M.: There is more to context than location: Environment sensing technologies for adaptive mobile user interfaces. In: Workshop on Interactive Applications of Mobile Computing (IMC'98). (1998)

[21] Dey, A.K.: Providing architectural support for building context–aware applications. PhD thesis, Georgia Institute of Technology (2000)

[22] Mäntyjärvi, J.: Sensor-based context recognition for mobile applications. PhD thesis, VTT Technical Research Centre of Finland (2003)

[23] Henricksen, K.: A Framework for Cotnext-Aware Pervasive Computing Applications. PhD thesis, School of Information Technology and Electrical Engineering at the University of Queensland (2003)

[24] Lieberman, H., Selker, T.: Out of context: Computer systems that adapt to, and learn from, context. In: IBM Systems Journal. Volume 39. (2000) 617–632

[25] Fitzpatrick, A., Biegel, G., Cahill, S.C.V.: Towards a sentient object model. In: Workshop on Engineering Context-Aware Object Oriented Systems and Environments (ECOOSE). (2002)

[26] Pascoe, J.: Adding generic contextual capabilities to wearable computers. In: Proceedings of the second International Symposium on Wearable Computers. (1998) 92–99

[27] Dey, A.K., Salber, D., Abowd, G.D., Futakawa, M.: The conference assistant: comtining context-awareness with wearable computing. In: Proceedings of the third International Symposium on Wearable Computers. (1999) 21–28

[28] Kortuem, G., Segall, Z., Bauer, M.: Context-aware, adaptive wearable computers as remote interfaces to 'intelligent' environments. In: Proceedings of the second International Symposium on Wearable Computers. (1998) 58–65

[29] Chen, G.: Solar: Building A Context Fusion Network for Pervasive Computing. PhD thesis, Hanover, New Hampshire (2004)

[30] Schmidt, A.: Ubiquitous Computing – Computing in Context. PhD thesis, Lancaster University, UK (2002)

[31] Himberg, J.: From insights to innovations: data mining, visualisation, and user interfaces. PhD thesis, Helsinki University of Technology (2004)

[32] Himberg, J., Korpiaho, K., Mannila, H., Tikanmäki, J., Toivonen, H.: Time series segmentation for context recognition in mobile devices. In: Proceedings of the 2001 IEEE International Conference on Data Mining. (2001) 203–210

[33] Mäntyjärvi, J., Himberg, J., Huuskonen, P.: Collaborative context recognition for handheld devices. In: Proceedings of the first IEEE International Conference on Pervasive Computing and Communications (PerCom'03). (2003) 161–168

[34] Schilit, W.N.: A System Architecture for Context-Aware Mobile Computing. PhD thesis, Columbia University (1995)

[35] Dey, A.K., Abowd, G.D., Salber, D.: A context-based infrastructure for smart environments. In: Proceedings of the first International Workshop on Managing Interactions in Smart Environments (MANSE'99). (1999) 114–128

[36] Schmidt, A., Laerhoven, K.V., Strohbach, M., Friday, A., Gellersen, H.W.: Context acquisition based on load sensing. In: Proceedings of Ubicomp 2002, Lecture Notes in Computer Science. Volume 2498., Springer Verlag (2002) 333 – 351

[37] Jacob, R.J., Ishii, H., Pangaro, G., Patten, J.: A tangible interface for organising information using a grid. In: Conference on Human Factors in Computing Systems (CHI 2002). (2002)

[38] Kidd, C.D., Orr, R., Abowd, G.D., Atkeson, C.G., Essa, I.A., MacIntyre, B., Mynatt, E., Starner, T.E., Newstetter, W.: The aware home: A living laboratory for ubiquitous computing research. In: Lecture in Computer Science; Proceedings of the Cooperative Buildings Integrating Information, Organisation and Architecture (CoBuild'99). Volume 1670. (1999)

[39] Ferscha, A., Hechinger, M., Mayrhofer, R., Oberhauser, R.: A peer-to-peer light-weight component model for context-aware smart space applications. In: International journal of wireless and mobile computing (IJWMC'04), special issue on Mobile Distributed Computing. (2004)

[40] Ferscha, A., Hechinger, M., Mayrhofer, R., dos Santos Rocha, M., Franz, M., Oberhauser, R.: Digital aura. In: Advances in Pervasive Computing, part of the 2nd International conference on pervasive computing (Pervasive 2004). Volume 176. (2004) 405–410

[41] Ferscha, A., Vogl, S., Beer, W.: Context sensing, aggregation, representation and exploitation in wireless networks. In: Scalable computing: Practice and experience. Volume 2. (2005) 77–81

[42] Ferscha, A., Resmerita, S., Holzmann, C., Reichör, M.: Orientation sensing for gesture-based interaction with smart artifacts. In: Computer communications. Volume 13. (2005)

[43] van Laerhoven, K., Cakmakci, O.: What shall we teach our pants? In: Proceedings of the 4th Symposium on Wearable Computers (ISWC'00). (2000) 77–83

[44] Gellersen, H.W., Schmidt, A., Beigl, M.: Multi-sensor context-awareness in mobile devices and smart artefacts. In: Mobile Networks and Applications. Volume 5. (2002) 341–351

[45] Randell, C., Muller, H.: The well mannered wearable computer. In: Personal and Ubiquitous Computing. (2002) 31–36

[46] Randell, C., Muller, H.: Context awareness by analysing accelerometer data. In: Proceedings of the fourth International Symposium on Wearable Computers. (2001) 175–176

[47] Randell, C., Muller, H.: Low cost indoor positioning system. In: Proceedings of Ubicomp 2001. (2001) 42–48

[48] Bell, M., Chalmers, M., Barkhuus, L., Hall, M., Sherwood, S., Tennet, P., Brown, B., Rowland, D., Benford, S., Hampshire, A., Capra, M.: Interweaving mobile games with everyday life. In: Proceedins of the 2006 ACM Conference on Human Factors in Computing Systems (CHI'06). (2006)

[49] Benford, S., Crabtree, A., Reeves, S., Flintham, M., Drozd, A., Sheridan, J., Dix, A.: The frame of the game: Blurring the boundary between fiction and reality in mobile experiences. In: Proceedins of the 2006 ACM Conference on Human Factors in Computing Systems (CHI'06). (2006) 427–436

[50] Benford, S., Bowers, J., Chandler, P., Ciolfi, L., Flintham, M., Fraser, M., Greenhalgh, C., Hall, T., Hellström, S.O., Izadi, S., Rodden, T., Schnädelbach, H., Taylor, I.: Unearthing virtual history: using diverse interfaces to reveal hidden virtual worlds. In: Proceedings of Ubicomp 2001. (2001) 1–6

[51] Brown, B., Randell, R.: Building a context sensitive telephone: Some hopes and pitfalls for context sensitive computing. Technical report, Equator (2002)

[52] Cakmakci, O., Coutaz, J., Laerhoven, K.V., Gellersen, H.W.: Context awareness in systems with limited resources. In: Proceedings of the 3rd Workshop on Artificial Intelligence in Mobile Systems (AIMS). (2002) 21–29

[53] Laerhoven, K.V., Schmidt, A., Gellersen, H.W.: Multi-sensor context aware clothing. In: Proceedings of the sixth International Symposium on Wearable Computers (ISWC). (2002) 49–57 14 Accelerometers but no further sensors.

[54] Brudna, C., Cahill, V., Casimiro, A., Cunningham, R., Kaiser, J., Martins, P., Meier, R., Sousa, P., Verissimo, P.: Final definition of cortex system architecture. available at http://www.dsg.cs.tcd.ie/sites/Cortex.html (2004)

[55] Satyanarayanan, M.: Scalable, secure, and highly available distributed file access. In: IEEE Computer. Volume 23. (1990) 9–18, 20–21

[56] Narayanan, D., Flinn, J., Satyanarayanan, M.: Using history to improve mobile application adaptation. In: Proceedings of the Third IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'00). (2000) 31

[57] Judd, G., Steenkiste, P.: Providing contextual information to pervasive computing applications. In: Proceedings of the first IEEE International Conference on Pervasive Computing and Communications (PerCom'03). (2003)

[58] Wang, H.J., Raman, B., nee Chuah, C., Biswas, R., Gummadi, R., Hohlt, B., Hong, X., Kiciman, E., Mao, Z., Shih, J.S., Subramanian, L., Zhao, B.Y., Joseph, A.D., Katz, R.H.: Iceberg: An internet-core network architecture for integrated communications. In: IEEE personal communications (special issue on IP-based mobile telecommunication networks). (2000)

[59] Raman, B., Wang, H.J., Shih, J.S., Joseph, A.D., Katz, R.H.: The iceberg project: Defining the ip and telecom intersection. In: IT Professional. (1999)

[60] Rhea, S., Eaton, P., Geels, D., Weatherspoon, H., Zhao, B., Kubiatowicz, J.: Pond: the oceanstore prototype. In: Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST'03). (2003)

[61] Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C., Zhao, B.: Oceanstore: An architecture for global-scale persistent storage. In: Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'00). (2000)

[62] Abowd, G.D., Dey, A.K., Brown, P.J., Davies, N., Smith, M., Steggles, P.: Towards a better understanding of context and context-awareness. In: HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, London, UK, Springer-Verlag (1999) 304–307

[63] Mayrhofer, R.M., Radi, H., Ferscha, A.: Recognising and predicting context by learning from user behaviour. In: The International Conference On Advances in Mobile Multimedia (MoMM2003). Volume 171. (2003) 25–35

[64] Brooks, R.A.: Elephants don't play chess. In: Robotics and Autonomous Systems. Volume 6. (1990)

[65] Padovitz, A., Bartolini, C., Zaslavski, A., Loke, S.W.: Extending the context space approach to management by business objectives. In: 12th Workshop of the HP OpenView University Association. (2005)

[66] Padovitz, A., Loke, W.W., Zaslavsky, A.: On uncertainty in context-aware computing: Appealing to high-level and same-level context for low-level context verification. In: Proceedings of the 1st International Workshop on Ubiquitous Computing (IWUC'04). (2004) 62–72

[67] Padovitz, A., Loke, S.W., Zaslavsky, A., Burg, B.: Towards a general approach for reasoning about context, situations and uncertainty in ubiquitous sensing: Putting geometrical intuitions to work. In: 2nd International Symposium on Ubiquitous Computing Systems (UCS'04). (2004)

[68] Rowling, J.: Harry Potter and the Prisoner of Azkaban. Bloomsbury publishing (2000)

[69] Chun, S.H., Kim, S.H.: Impact of momentum bias on forecasting through knowledge discovery techniques in the foreign exchange market. In: Expert Systems with Applications. Volume 24. (2003) 115–122

[70] Chun, S.H., Kim, S.H.: Data mining for financial prediction and trading: application to single and multiple markets. In: Expert Systems with Applications. Volume 26. (2004) 131–139

[71] Horvitz, E., paul Koch, Kadie, C.M., Jacobs, A.: Coordinate: Probabilistic forecasting of presence and availability. In: Proceedings of the Eighteenth Conference on Uncertainty and Artificial Intelligence. (2002) 224–233

[72] Brown, P., Burleson, W., Lamming, M., Rahlff, O.W., Romano, G., Scholtz, J., Snowdon, D.: Context-awareness: Some compelling applications. In: Proceedings of the CH12000 Workshop on the What, Who, Where, When, Why and How of Context-Awareness. (2000)

[73] Mozer, M.C.: The neural network house: An environment that adapts to its inhabitants. In: Proceedings of the American Association for Artificial Intelligence Spring Symposium on Intelligent Environments. (1998) 110–114

[74] Sigg, S., Haseloff, S., David, K.: Minimising the context prediction error. In: 65th IEEE semi-annual Vehicular Technology Conference (VTC2007-Spring) (to appear). (2007)

[75] Leichtenstern, K., Luca, A.D., Rukzio, E.: Analysis of built-in mobile phone sensors for supporting interactions with the real world. In: Pervasive Mobile Interaction Devices (PERMID 2005) - Mobile Devices as Pervasive User Interfaces and Interaction Devices - Workshop at the Pervasive 2005. (2005)

[76] Mulvenna, M., Nugent, C., Gu, X., Shapcott, M., Wallace, J., Martin, S.: Using context prediction for self-management in ubiquitous computing environments. In: Consumer Communications and Networking Conference (CCNC). (2006) 600–604

[77] Brown, P.J., Jones, G.J.F.: Exploiting contextual change in context–aware retrieval. In: Proceedings of the 2002 ACM Symposium on Applied Computing. (2002) 650–656

[78] Cottrell, M., de Bodt, W., Gregoire, P.: Simulating interest rate structure evolution on a long term horizon: A kohonen map application. In: Proceedings of Neural Networks in the Capital Markets. (1996)

[79] Silc, J., Robic, B., Ungerer, T.: Processor Architecture - From Dataflow to Superscalar and Beyond. Springer-Verlag (1999)

[80] Vintan, L., Gellert, A., Petzold, J., Ungerer, T.: Person movement prediction using neural networks. Technical report, Institute of Computer Science, University of Augsburg (2004)

[81] Petzold, J., Bagci, F., Trumler, W., Ungerer, T.: Global and local state context prediction. In: Artificial Intelligence in Mobile Systems (AIMS 2003) in Conjunction with the Fifth International Conference on Ubiquitous Computing. (2003)

[82] Petzold, J., Pietzowski, A., Bagci, F., Trumler, W., Ungerer, T.: Prediction of indoor movements using bayesian networks. In: First International Workshop on Location- and Context-Awareness (LoCA 2005). (2005)

[83] Petzold, J., Pietzowski, A., Bagci, F., Trumler, W., Ungerer, T.: Confidence estimation of the state predictor method. In: 2nd European Symposium on Ambient Intelligence. (2004)

[84] Petzold, J., Pietzowski, A., Bagci, F., Trumler, W., Ungerer, T.: The state predictor method for context prediction. In: Adjunct Proceedings Fifth International Conference on Ubiquitous Computing. (2003)

[85] Jursa, R., Lange, B., Rohrig, K.: Advanced wind power predicting with artificial intelligence methods. In: Artificial Intelligence In Energy Systems and Power (AIESP 2006). (2006)

[86] Davison, B.D., Hirsh, H.: Predicting sequences of user actions. In: AAAI/ICML Workshop on Predicting the Future: AI Approaches to Time–Series Analysis. (1998)

[87] Weiss, G.M., Hirsh, H.: Learning to predict rare events in categorical time-series data. In: Predicting the future: ai approaches to time-series problems; Workshop in conjunction with the fifteenth national conference on artificial intelligence. (1998) 83–90

[88] Gorniak, P., Poole, D.: Predicting future user actions by observing unmodified applications. In: Conference of the American Association for Artificial Intelligence. (2000)

[89] Feder, M., Merhav, N., Gutman, M.: Universal prediction of individual sequences. IEEE Transactions on Information Theory **38**(4) (1992) 1258–1270

[90] Simon, G., Lendasse, A., Cottrell, M., Fort, J.C., Verleysen, M.: Time series forecasting: Obtaining long term trends with self-organising maps. Pattern Recognition Letters **26**(12) (2005) 1795–1808

[91] Cottrell, M., Fort, J.C., Pages, G.: Theoretical aspects of the som algorithm. In: Neurocomputing. Volume 21. (1998) 119–138

[92] Walter, J., Ritter, H., Schulten, K.: Non-linear prediction with self-organising maps. In: Proceedings of IJCNN. (1990) 589–594

[93] Vesanto, J.: Using the som and local models in time series prediction. In: Proceedings of Workshop on Self-Organising Maps (WSOM'97). (1997) 209–214

[94] Koskela, T., Varsta, M., Heikkonen, J., Kaski, K.: Recurrent som with local linear models in time series prediction. In: European Symposium on Artficial Neural Networks. (1998) 167.172

[95] Lendasse, A., Verleysen, M., de Bodt, E., Cottrell, M., Gregoire, P.: Forecasting time-series by kohonen classification. In: European Symposium on Artificial Neural Networks. (1998) 221–226

[96] Verleysen, M., de Bodt, E., Lendasse, A.: Forecasting financial time series through intrinsic estimation and non-linear data projection. In: Lecture Notes in Computer Science; Proceedings of the International Workconference on Artificial and Natural Neural Networks. Volume 1607. (1999)

[97] Kaowthumrong, K., Lebsack, J., Han, R.: Automated selection of the active device in interactive multi-device smart spaces. In: Spontaneity Workshop of Ubicomp 2002. (2002)

[98] Papakyriazis, A., Papakyriazis, P.: Adaptive prediction: a sequential approach to forecasting and estimation of nonstationary environmental systems. In: Kybernetes. Volume 28. (1999)

[99] Patterson, D.J., Liao, L., Fox, D., Kautz, H.: Inferring high-level behaviour from low-level sensors. In: 5th international Conference on Ubiquitous Computing (UbiComp). Volume 5. (2003) 73–89

[100] Laasonen, K., Raento, M., Toivonen, H.: Adaptive on-device location recognition. Number 3001 in LNCS (2004) 287–304

[101] Mannila, H., Toivonen, H., Verkamo, A.: Discovery of frequent episodes in event sequences. In: Data Mining and Knowledge Discovery, Kluwer Academic Publishers (1997) 259–289

[102] Horvitz, E., Jacobs, A., Hovel, D.: Attention-sensitive alerting. In: Proceedings of UAI'99; conference on Uncertainty and Artificial Intelligence. (1999) 305–313

[103] Cook, D.J., Youngblood, M., Heierman, E., Gopalratnam, K., Rao, S., Litvin, A., Khawaja, F.: Mavhome: An agent–based smart home. In: 1st IEEE International Conference on Pervasive Computing and Communications (PerCom?03), IEEE Computer Society Press (2003) 521–524

[104] Gopalratnam, K., D.J.Cook: Active lezi: An incremental parsing algorithm for sequential prediction. In: Proceedings of the Florida Artificial Intelligence Research Symposium. (2003)

[105] Heierman, E.O., Cook, D.J.: Improving home automation by discovering regularly occurring device usage patterns. In: ICDM. (2003) 537–540

[106] Boeckenhauer, H.J., Bongartz, D.: Algorithmische Grundlagen der Bioinformatik. Teubner (2003) (in German).

[107] Brockwell, J., Davis, R.: Introduction to Time-Series and Forecasting. Springer (1996)

[108] Sigg, S., Haseloff, S., David, K.: A novel approach to context prediction in ubicomp environments. In: Proceedings of the 17th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2006). (2006)

[109] Barkhuus, L.: How to define the communication situation: Context measures in present mobile telephony. In: Context, Stanford, CA, Springer (2003)

[110] Greenberg, S.: Context as a dynamic construct. In: Human-Computer Interaction. Volume 16 (2-4)., awrence Erlbaum Associates Inc. (2001) 257–268

[111] Anderson, J.R.: Cognitive psychology and its implications. 3 edn. Spektrum (2001)

[112] Magnusson, M.S.: Repeated patterns in behavior and other biological phenomena. In Oller, K., Griebel, U., eds.: Evolution of Communication Systems: A Comparative Approach. MIT Press, Cambridge, MA (2004) 111–128

[113] Jonsson, G.K., Bjarkadottir, S.H., Gislason, B., Borrie, A., Magnusson, M.S.: Detection of real-time patterns in sports: interactions in football. In: L'ethologie applique aujourd'hui. Volume 3., C. Baudoin (2003)

[114] Krsul, I.: Authorship analysis: Identifying the author of a program. Technical report, Department of Computer Sciences, Purdue University (1994)

[115] Magnusson, M.S.: Understanding social interaction: Discovering hidden structure with model and algorithms. In: The Hidden Structure of Interaction: From Neurons to Culture Patterns, L.Anolli, S.Duncan Jr., M.S.Magnusson and G.Riva (2005)

[116] Kreiss, J.P., Neuhaus, G.: Einführung in die Zeitreihenanalyse (in German). Springer-Verlag (2006)

[117] Duda, R., Hart, P., Stork, D.: Pattern Classification. 2nd edn. Wiley Interscience (2001)

[118] Wooldridge, M., Jennings, N.R.: Intelligent agents: Theory and practice. In: Knowledge Engineering Review. Volume 10. (1995)

[119] Box, G.E.P., Jenkins, G.M.: Time Series Analysis forecasting and control. Holden-Day (1976)

[120] Wegener, I.: Theoretische Informatik – eine algorithmenorientierte Einführung. Volume 2nd. B.G.Teubner (1999)

[121] Ashbrook, D., Starner, T.: Learning significant locations and predicting user movement with gps. (2002)

[122] Ferscha, A.: Pervasive computing. Datenbank-Spektrum (2003) 48–51

[123] Moore, T.: Life of Lord Byron (With his letters and journals). Indypublish (2005)

[124] Altschul, S.F., Madden, T.L., Schäfer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.: Gapped blast and psi–blast: a new generation of protein database search programs. In: Nucleic Acids Research. Volume 25. (1997) 3389–3402

[125] Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. In: Journal for molecular biology. Volume 215. (1990) 403–410

[126] Pearson, W.R.: Flexible sequence similarity searching with the fasta3 program package. In: Methods in Molecular Biology: Bioinformatics Methods and Protocols). Volume 132. (2000) 185–219

[127] Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology **48**(3) (1970) 443–453

[128] Jerome, J.K.: Three men in a boat. Collector's Library (2005)

[129] Loeffler, T., Sigg, S., Haseloff, S., David, K.: The quick step to foxtrot. In David, K., Droegehorn, O., Haseloff, S., eds.: Proceedings of the Second Workshop on Context Awareness for Proactive Systems (CAPS 2006), Kassel University press (2006)

[130] Harries, M., Horn, K., Sammut, C.: Learning in time ordered domains with hidden changes in context. In: Predicting the future: AI approaches to time-series problems. (1998) 29–33

[131] Norman, D.: The invisible computer. MIT press (1999)

[132] Droste, S., Jansen, T., Wegener, I.: Perhaps not a free lunch but at least a free appetiser. Technical report, University of Dortmund, Sonerforschungsbereich (SFB) 531 (1998)

[133] Keogh, E.J., Pazzani, M.J.: An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In: Predicting the future: ai approaches to time-series problems; Workshop in conjunction with the fifteenth national conference on artificial intelligence. (1998) 44–51

[134] Vapnik, V.N.: Estimation of Dependencies Based on Empirical Data (in Russian). Nauka (1979) (English translation: Springer Verlag, New York, 1982).

[135] Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer (2000)

[136] Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery **2**(2) (1998) 121–167 Helpful tutorial.

[137] S.Russell, P.Norvig: Artificial Intelligence A Modern Approach. 2nd edn. Prentice Hall (2003)

[138] Gold, C., Holub, A., Sollich, P.: Bayesian approach to feature selection and parameter tuning for support vector machine classifiers. Neural Netw. **18**(5-6) (2005) 693–701

[139] Schwaighofer, A., Tresp, V.: The bayesian committee support vector machine. **2130** (2001) 411–417

[140] Feller, W.: An Introduction to Probability Theory and its Applications. Wiley (1968)

[141] Kohonen, T.: Self-organised formation of topologically correct feature maps. In: Biological Cybernetics. Number 43 (1982) 59–69

[142] Kohonen, T.: Analysis of a simple self-organising process. In: Biological Cybernetics. Number 43 (1982) 59–69

[143] Kohonen, T.: Self-organisation and Associative Memory. 3rd edn. Springer (1984)

[144] Kohonen, T.: Self-Organising Maps. Volume 30. Springer (1995)

[145] Cottrell, M., Fort, J., Pages, G.: Theoretical aspects of the som algorithm. In: Neurocomputing. Volume 21. (1998) 119–138

[146] Cottrell, M., Girard, B., Rousset, P.: Forecasting of curves using a kohonen classification. In: Journal of Forecasting. Number 17 (1998) 429–439

[147] Hsu, W.H., Gettings, N.D., Lease, V.E., Pan, Y., Wilkins, D.C.: Heterogeneous time series learning for crisis monitoring. In: Predicting the future: ai approaches to time-series problems. Workshop held in conjunction with the fifteenth national conference on artificial intelligence. Volume 98. (1998) 34–41

[148] Mozer, M.C.: Neural net architectures for temporal sequence processing. In Weigend, A.S., Gershenfeld, N.A., eds.: Predicting the Future Understanding the Past, Addison Wesley (1994)

[149] Chatfild, C.: The Analysis of Time Series: An Introduction. Volume 5. Chapman and Hall (1996)

[150] Mehrotra, K., Mohan, C.K., Ranka, S.: Elements of Artificial Neural Networks. MIT Press (1997)

[151] Cadzow, J., Ogino, K.: Adaptive arma spectral estimation. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'81). Volume 6. (1981) 475–479

[152] Capra, L., Musolesi, M.: Autonomic trust prediction for pervasive systems. In: AINA '06: Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA'06), Washington, DC, USA, IEEE Computer Society (2006) 481–488

[153] Musolesi, M., Mascolo, C.: Evaluating context information predictability for autonomic communication. In: Proceedings of 2nd IEEE Workshop on Autonomic Communications and Computing (ACC'06). Co-located with 7th IEEE Int. Symp. WoWMoM'06, Niagara Falls, NY, IEEE Computer Society Press (2006)

[154] Chapman, C., Musolesi, M., Emmerich, W., Mascolo, C.: Predictive Resource Scheduling in Computational Grids. In: Proceedings of the 21^{st} International Parallel and Distributed Processing Symposium, Long Beach, CA, IEEE Computer Society Press (2007)

[155] Goris, M.J., Gray, D.A., Mareels, I.M.: Reducing the computational load of a kalman filter. In: IEE Electronics Letters. (1997)

[156] Adams, D.: Mostly harmless. Rel Dey (2000)

[157] Petzold, J., Bagci, F., Trumler, W., Ungerer, T.: Next location prediction within a smart office building. In: 1st International Workshop on Exploiting Context Histories in Smart Environments (ECHISE) at the 3rd International Conference on Pervasive Computing. (2005)