Vorlesung: Algorithms for context prediction in ubiquitous systems

Wintersemester 2008/09

Version vom November 18, 2008

Veranstalter: Stephan Sigg

Technische Universität Braunschweig
Institut für Betriebssysteme und Rechnerverbund
Verteilte und Ubiquitäre Systeme

D-38106 Braunschweig

Contents

Abbreviations and Notation

The following figures and notations are utilised throughout this document. It has been attempted to keep the standard notation from the literature whenever possible. However, since the document touches diverse scientific areas, the notation had to be adapted 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	99
AR	Autoregressive	94
ARMA	Autoregressive Moving average	39
BN	Bayesian Net	73
C	A set of context elements	80
c_i	A context element	21
χ_i	Observation or event of a stochastic process	40
d_i	Actual value of the context element at time t_{0+i}	41
δ	Error threshold of one context source	??
DIM	Time series dimension	??
$dim(T)$	The dimension of time series T	33
ETH	Swiss Federal Institute of Technology Zürich, Switzerland	??
f	Numerical functions are denoted with f	34
$G = (V, E)$	A graph G with vertices in V and edges in E	??
GPS	Global Positioning System	9
GSM	Global System for Mobile Communications	9

Notation	Explanation	Page
ID	Identification	21
ISET	Institut für solare Energieversorgungstechnik, Kassel, Germany	??
IST	Priority of the 6th framework programme of the European Union: Information Society Technology	??
k	Context history size	36
KM	Kernel Machines	73
λ	An empty time series.	33
LEN	Context history size	??
M	Matrices are denoted with M	88
m	Number of context sources and low-level contexts in one time interval. Dimension of low-level context time series.	53
MEG	Mobile Event Guide	??
MIT	Massachusetts Institute of Technology	18
MM	Markov Models	73
n	Length of the prediction horizon	41
NN	Neural Nets	73
NNS	Nearest Neighbour Search	73
o	Number of context high-level contexts in one time interval. Dimension of high-level context time series.	53
P_{acq}	Probability that no error occurs in the context acquisition process	53
P_{hl}	Probability that context prediction based on high-level context elements is without error	55
p_i	Outcome of the prediction process of the context element at time t_{0+i}	41
P_{int}	Probability that no error occurs in the context interpretation process	53
P_{ll}	Probability that context prediction based on low-level context elements is without error	56
$P_{ll}(i), P_{hl}(i)$	Probability that the prediction based of the i-th context element is without error for low-level and high-level context prediction schemes respectively.	55, 56
P_{pre}	Probability that no error occurs in the context prediction process	53
π	A stochastic process	40
PM	Pattern Matching	73
PyS60	Python for Symbian Series 60 platform	??
RMSE	Root of the Mean Squared Error	??

Notation	Explanation	Page
S	A search space	??
S_l, S_h	Search spaces of high-level and low-level context prediction schemes	??
S60	Symbian Series 60	??
SOM	Self Organising Map	??
SPM	State Predictor Method	99
SVM	Support Vector Machine	73
T, T'	Time series	33
T_{t_j,t_k}	Time series T in the interval $[t_j, t_k]$	33
$\|T\|$	Number of time series elements in time series T	33
t_i	A time interval	21
τ	A learning threshold	??
TecO	Telecooperation Office Karlsruhe, Germany	??
noalign TS	Time series	??
UbiComp	Ubiquitous Computing	20
UMTS	Universal Mobile Telecommunications System	9
v_i	Measured context value at time t_{0+i}	??
$\overrightarrow{v}$	A vector $v = (v_1, \ldots, v_\kappa)$	85
v_l	Number of legal values for low-level context time series elements	53
v_h	Number of legal values for high-level context time series elements	53
WLAN	Wireless Local Area Network	9
ξ_i	Time series element	33

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.

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 [7])

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 [8] 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 [9]

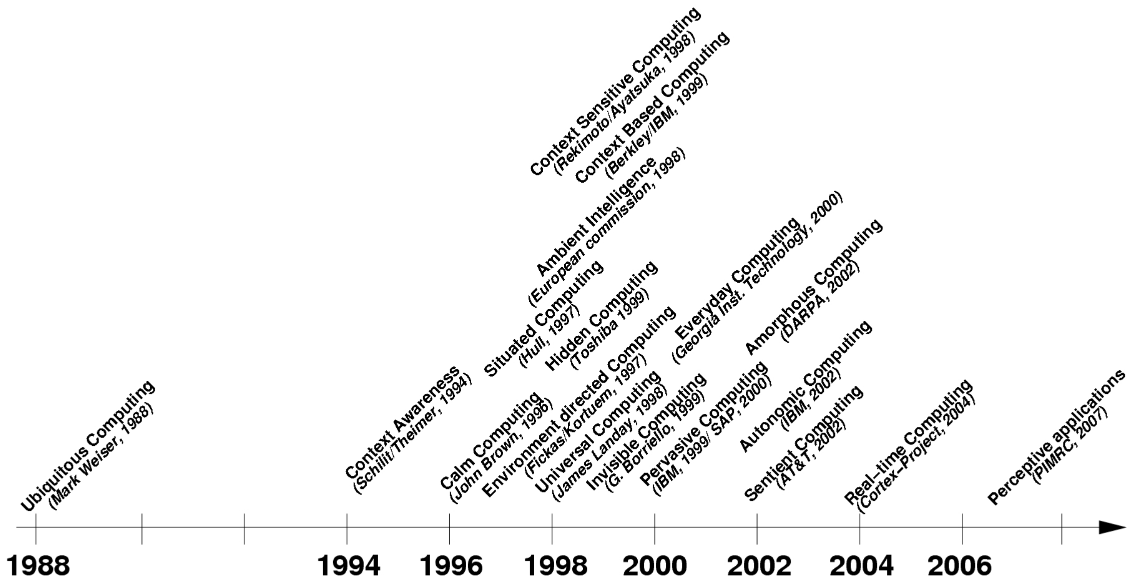


Figure 2.1: Concepts related to Ubiquitous computing

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 [10]. Following this pioneering work, numerous further concepts and ideas have been discussed by various research groups.

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 [11]. 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.

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 [10] 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"[12] 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 [13]. 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 [14]. These categories are 'user context', 'physical context' and 'computing context'. As further categories, Brown added information about the time [15], while Pascoe also considered the blood pressure of users [16]. Dey took the latter proposal to a broader scope by considering emotions and the focus of attention [17].

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 [18]. 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 [19].

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 [20], 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 [21]. 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 [22]. Other projects refine the definition of context to their individual needs. In [23] 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 [13] 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 [24]. The authors of [25] 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 [13, 15, 26].

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 [19].

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. [23]).

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 [19]. 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 [27]. 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 [28]. 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 [29]. 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 [20]. 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 [19]. 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 [30] 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 [31].

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 [32], 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 [19]. 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 [33].

Later on, Albrecht Schmidt presented a "working model for context-aware mobile computing" which is basically an extensible tree structure [28]. 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 [21]. 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 [27]. 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 [34]. 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

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

kitchen is considered that attempts to interpret peoples' intentions and reacts accordingly.

Additionally, the SenseBoard has been proposed in [35]. 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.

2.2 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 [11]. 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 [21] 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 3.1.3.

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 [18, 28]. 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 [19] 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 [21], 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 [36] 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

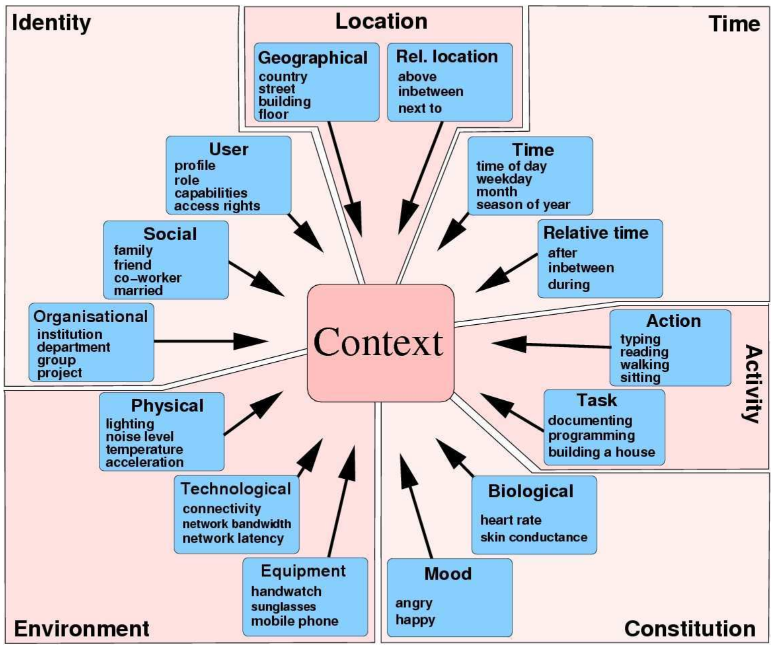


Figure 2.2: Aspects of context

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 in advance. Several aspects of context have been introduced in [6, 37]. 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 [38].

A rough distinction between low-level and higher level contexts is made by Anind K. Dey, Bill Schilit and Marvin Theimer [19, 13]. 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 [20]. 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.

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[2] (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

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

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[3]	GPS sensor
swimming	47° 25.5634'N; 007° 39.3538'E	GPGGA[4]	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

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.

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, 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

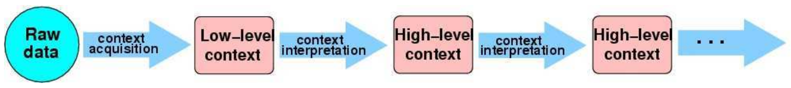


Figure 2.3: Context pre-processing steps.

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 4. Various context processing steps and corresponding input and output data are depicted in figure 2.3.

General assumptions

We assume 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. 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, 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.

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

Table 2.2 depicts the context data types of the context types introduced in figure 2.2[5].

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-

[5]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

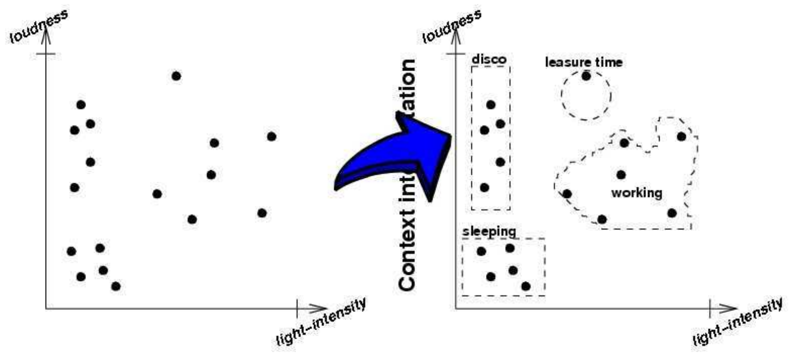


Figure 2.4: Illustration of the context interpretation step.

dimensional coordinate system. This representation has first been considered by Padovitz et al [39, 40, 41]. 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[6].

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 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 [41].

[6]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.

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 [42])

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 **??**). 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, [43, 44]).

However, the different notions become mixed as some authors also use the terms forecasting or proactivity in order to describe the context prediction process [45, 4]. Some authors even utilise these notions in order to describe the process of inferring a context [46, 47]. To make things even more complicated, the term context prediction is not used uniformly by researchers. While the authors in [6, 4, 48] employ the term context prediction to describe an operation that infers future contexts from past and present contexts, [46] uses this term in order to describe the automatic triggering of actions when some context becomes active, while the authors of [49, 50] 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 4. 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 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.

3.1.1 Time series and context patterns

Context prediction requires the consideration of the time dimension. A set of observations $\xi_{t_1} \dots \xi_{t_n}$ with ξ_{t_i} being recorded at a specific time interval t_i, is called a time series [51].

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 [15] suggest a classification of context-aware applications into continuous and discrete. 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 [52].

Definition 3.1.1 : 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 $c_{t_1} \dots c_{t_n}$ with the interval t_i starting before the interval t_{i+1}, the time series T_{t_2,t_n} covers the context elements $c_{t_2}, \dots, c_{t_n}$ 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 3.1.2 : 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 3.1.3 : 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.

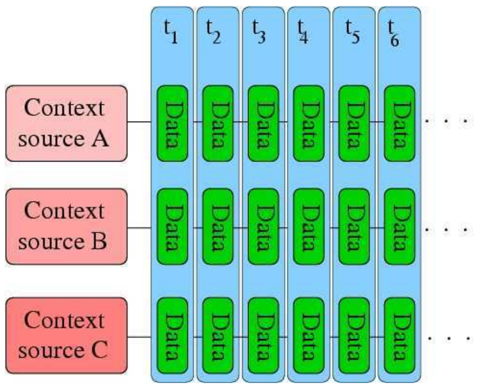


Figure 3.1: Illustration of a multidimensional time series.

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.

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.

Definition 3.1.4 : 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 3.1.3 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.

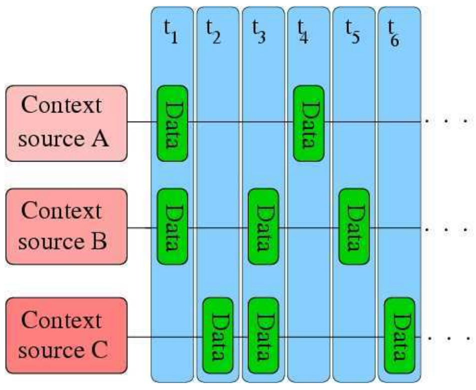


Figure 3.2: Illustration of a realistic time series.

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 $c_1 < \dots < c_\kappa$ which are in between considering the $<$-relation ($c < c_1 < \dots < c_k < c'$).

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 3.1.5 : 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 [53]. 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 3.1.6 : 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 $|T_{0-k,0}|$.*

3.1.2 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 [54]. Also, as the authors in [55] 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 [56]. 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 [56].

These findings can be observed in various fields. As [57] states, ”Behaviour consists of patterns in time”. The authors of [58] 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 [59]. Some work even draws a connection between behaviour patterns and patterns found in DNA-sequences [60]. For the remainder of this document we assume that typical patterns exist in human behaviour and that these patterns can be described by context sequences like context time series.

3.1.3 Challenges in UbiComp environments

Context prediction in ubiquitous computing environments is seriously affected by the heavily flexible computing environment [21]. 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.

3.1.4 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, 37], time series forecasting [61, 62] and event prediction [63], 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 [45, 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 [64]. This definition is too wide to accurately define the context prediction task.

In [39] 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 3.1.3, 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 [61, 51, 65]. 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 [63]. 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:

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

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 [66], a search problem is defined as follows.

Definition 3.1.7 : 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 3.1.8 : Prediction quality

> *Let T denote a time series and $d : T \times T \to \mathbb{R}$ 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 3.1.9 : 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 $f_{t_i} : T_{t_{i-k+1},t_i} \to T_{t_{i+1},t_{i+n}}$ that approximates $\pi(t)$.*

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

Definition 3.1.10 : 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 [67]. However, this definition is not applicable to context prediction in ubiquitous computing environments, since possibly incompletely sampled, multi-type time series are not covered.

3.1.5 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 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, 68, 69, 70, 4, 37, 71]). 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

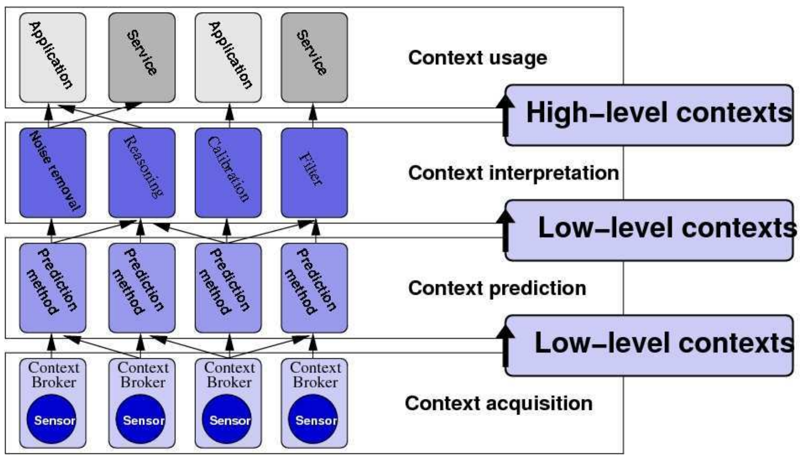


Figure 3.3: Context prediction based on low-level context elements.

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[1]. In the uppermost picture in figure 3.5, 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

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

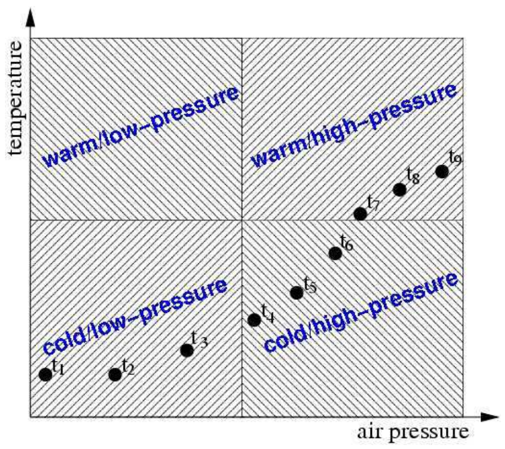


Figure 3.4: A set of measurements from context sources for temperature and air pressure.

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.

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.

We primarily focus on the accuracy of the overall context prediction procedure (cf. section 3.1.4). 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.

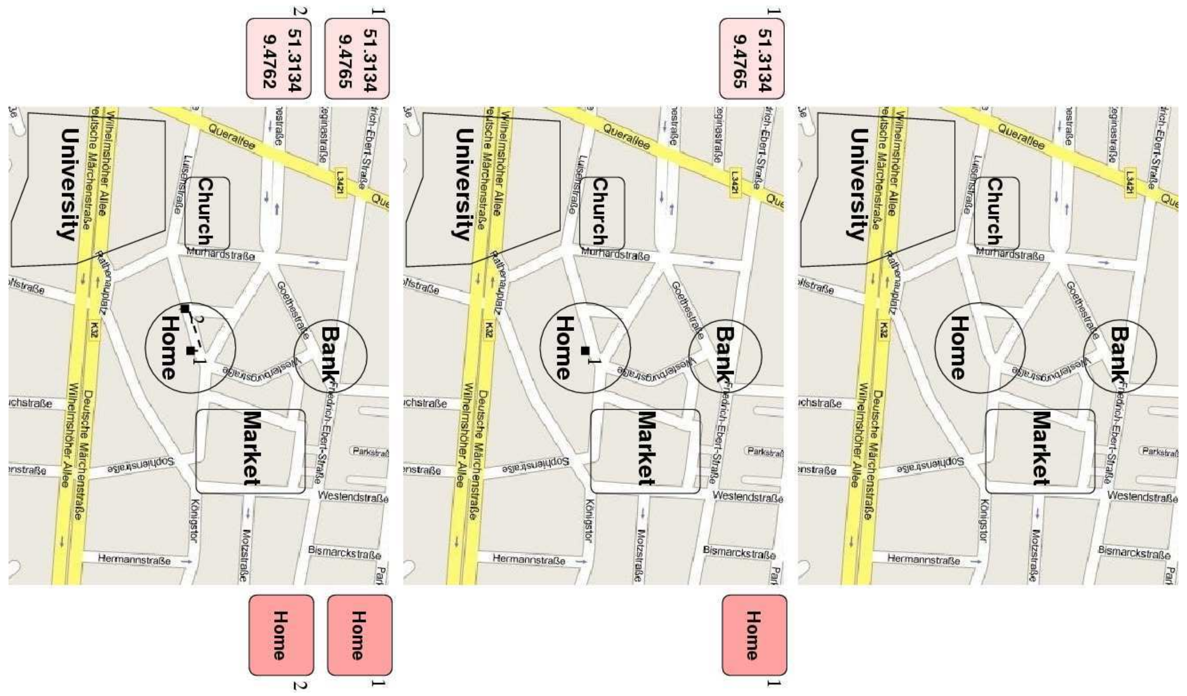


Figure 3.5: Movement of a user tracked by classified high-level contexts and by low-level contexts.

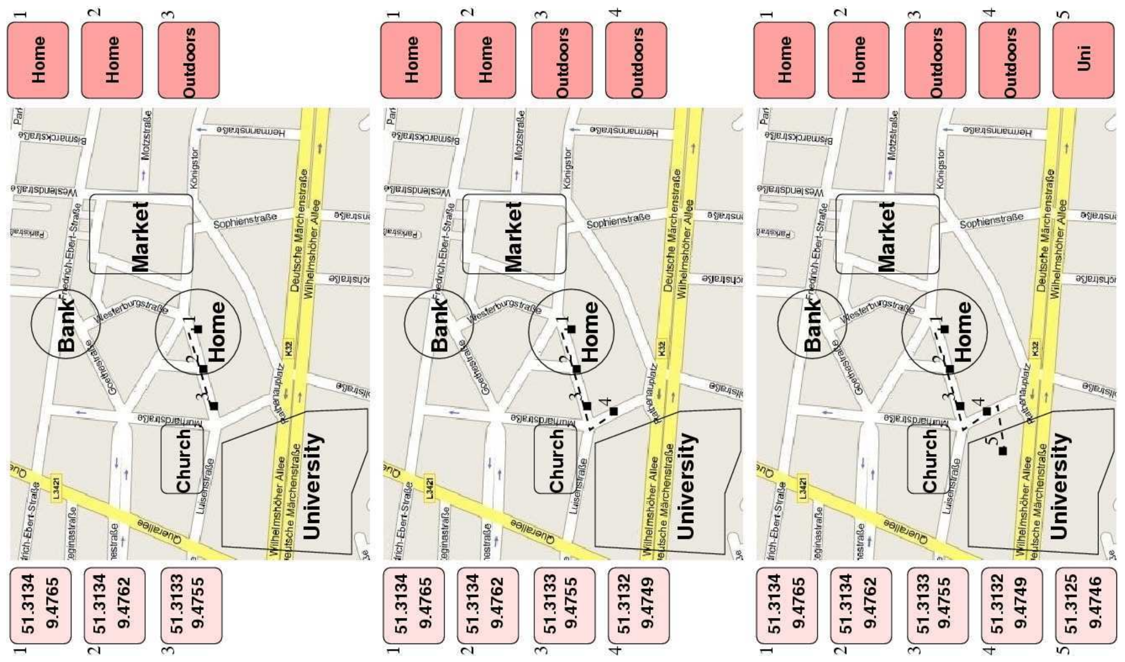


Figure 3.6: Movement of a user tracked by classified high-level contexts and by low-level contexts.

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 3.1.4.

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 [54], 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 **??**.

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.

4 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 [72])

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.

4.1 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 3.1.4).

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 4.1.

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 3.1.5). 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.

4.1.1 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 operations 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

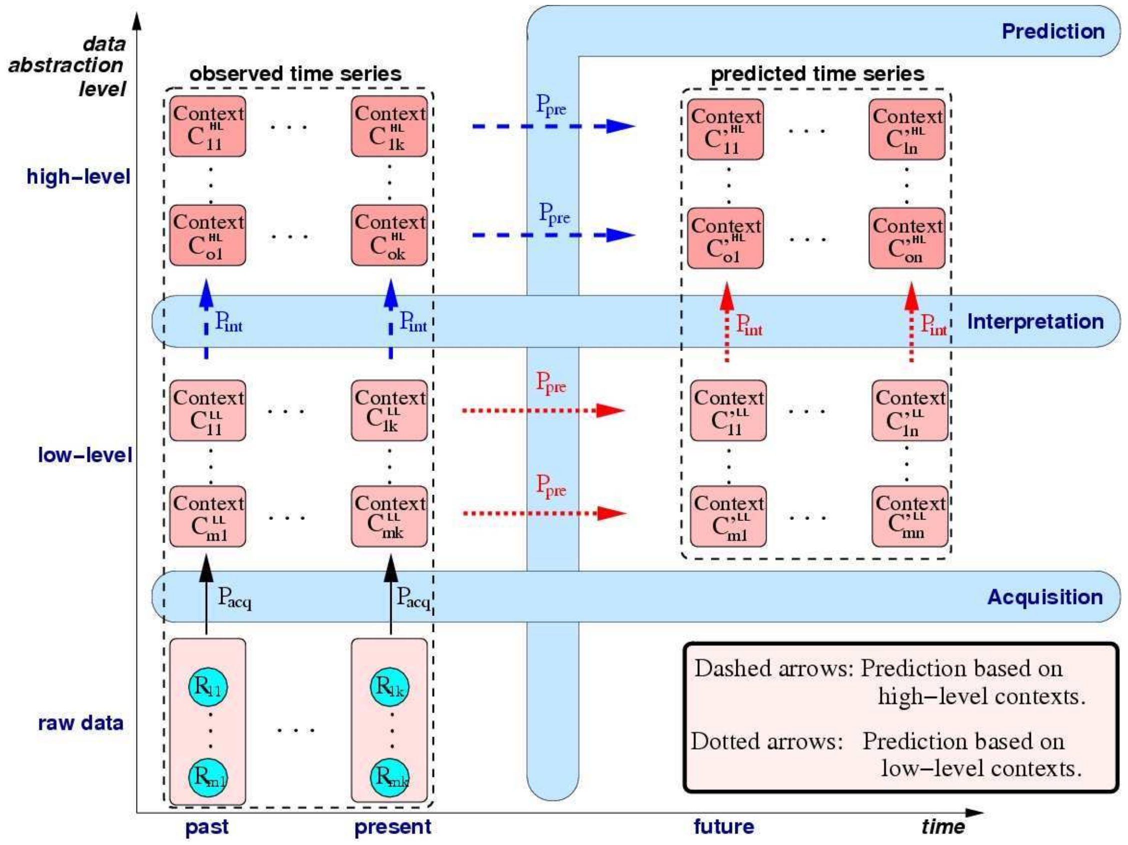


Figure 4.1: Low-level and high-level context prediction schemes.

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 4.1 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

$$P_{hl}^{approx} = P_{acq}^{km} P_{int}^{ko} P_{pre}^{o}(i) \tag{4.1}$$

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

$$P_{cor}^{int} = (1 - P_{acq}^{m})(1 - P_{int}^{o})\frac{1}{v_h^o - 1}. \tag{4.2}$$

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

$$\begin{aligned} P_{hl}(i) &= \left(P_{acq}^m P_{int}^o + P_{cor}^{int}\right)^k P_{pre}^o(i) \\ &+ \left(1 - \left(P_{acq}^m P_{int}^o + P_{cor}^{int}\right)^k\right) \frac{1 - P_{pre}^o(i)}{v_h^o - 1}. \end{aligned} \tag{4.3}$$

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

$$\begin{aligned} P_{hl} &= \left(P_{acq}^m P_{int}^o + P_{cor}^{int}\right)^k P_{pre}^o \\ &+ \left(1 - \left(P_{acq}^m P_{int}^o + P_{cor}^{int}\right)^k\right) \left(\frac{1 - P_{pre}}{v_h^n - 1}\right)^o. \end{aligned} \tag{4.4}$$

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

$$\left(1 - \left(P_{acq}^m P_{int}^o + P_{cor}^{int}\right)^k\right) \tag{4.5}$$

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

$$P_{cor}^{pre} = (1 - P_{acq}^k)(1 - P_{pre}(i))\frac{1}{v_l - 1} \tag{4.6}$$

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

$$\left(P_{acq}^k P_{pre}(i) + P_{cor}^{pre}\right)^m P_{int}^o \tag{4.7}$$

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

$$\begin{aligned} P_{ll}(i) &= \left(P_{acq}^k P_{pre}(i) + P_{cor}^{pre}\right)^m P_{int}^o \\ &+ \left(1 - \left(P_{acq}^k P_{pre}(i) + P_{cor}^{pre}\right)^m\right) \frac{1 - P_{int}^o}{v_h^o - 1}. \end{aligned} \tag{4.8}$$

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

$$\begin{aligned} P_{ll} &= \left(P_{acq}^k P_{pre} + \left(1 - P_{acq}^k\right)(1 - P_{pre}) \frac{1}{v_l^n - 1}\right)^m P_{int}^o \\ &+ \left(1 - \left(P_{acq}^k P_{pre} + \left(1 - P_{acq}^k\right)(1 - P_{pre}) \frac{1}{v_l^n - 1}\right)^m\right) \\ &\cdot \left(\frac{1 - P_{int}^o}{v_h^o - 1}\right)^n. \end{aligned} \tag{4.9}$$

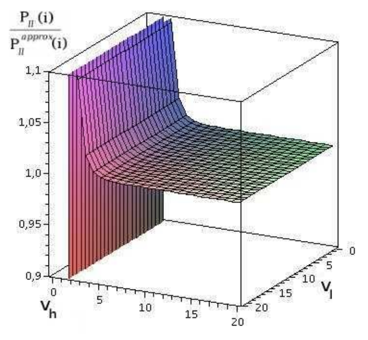
Discussion

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

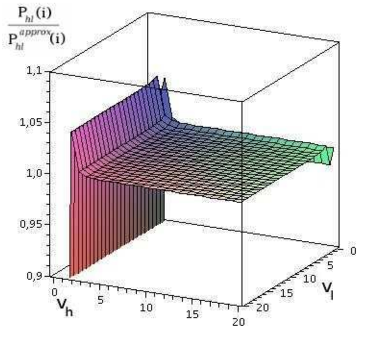
$$P_{ll}^{approx}(i) = P_{acq}^{km} P_{pre}^m(i) P_{int}^o \tag{4.10}$$

$$P_{hl}^{approx}(i) = P_{acq}^{km} P_{pre}^o(i) P_{int}^{ko}. \tag{4.11}$$

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



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



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

Figure 4.2: Comparison of approximated to exact probability of prediction errors for $k = m = o = 5$ and $P_{acq} = 0.99, P_{int} = P_{pre}(i) = 0.9$.

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 4.2 shows $\frac{P_{ll}(i)}{P_{ll}^{approx}(i)}$ and $\frac{P_{hl}(i)}{P_{hl}^{approx}(i)}$ respectively for $k = m = o = 5, P_{acq} = 0.99, P_{pre}(i) = 0.9, P_{int} = 0.9$.

These parameters are chosen to represent settings in typical scenarios. But of course, these results can not be generalised to all parameter settings.

From figure 4.2 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, livin-groom 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

$$\frac{P_{hl}^{approx}(i)}{P_{ll}^{approx}(i)} = P_{int}^{k} P_{pre}^{o-m}(i). \tag{4.12}$$

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.

Regarding the exact probabilities $P_{hl}(i)$ and $P_{ll}(i)$ we summarise the results in the following. A detailed discussion is provided in [73].

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 4.3 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 4.4 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.

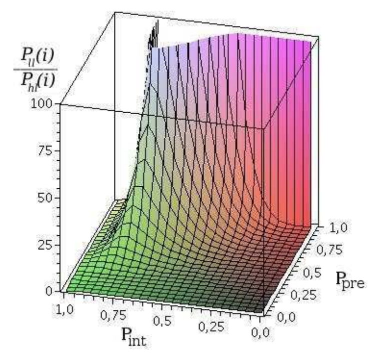
Observe that the high-level context prediction scheme is only superior for values of $P_{pre}(i)$ that are below 0.25. We argue that these low values for the prediction process are not acceptable for any utilisation of context prediction in real world scenarios.

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.

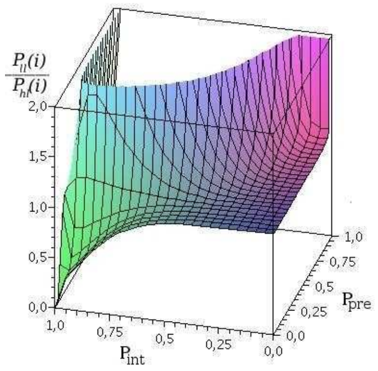
We have observed therefore that the high-level context prediction scheme has the higher probability to compute inaccurate context predictions even for environments where the context time series dimensionality as well as the context history size are small. With increasing size of the environment or scenario 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.

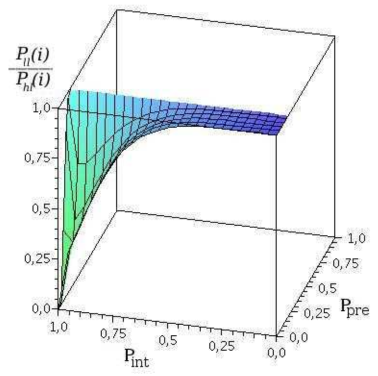


(a) $\frac{P_{ll}(i)}{P_{hl}(i)}$ in the range of 0 to 100.

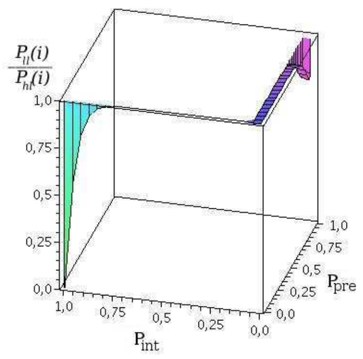


(b) $\frac{P_{ll}(i)}{P_{hl}(i)}$ in the range of 0 to 2.

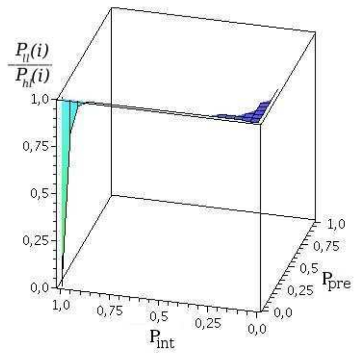
Figure 4.3: Comparison of the low-level and high-level context prediction schemes.



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

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 Basics on probability theory

The methods dicussed in later chapters require some basic knowledge on the notion of mathematical probability. We can differentiate between probability spaces with finite or infinite number of events. For this lecture it suffices to consider only probability spaces with finite events. In context awareness we heavily rely on sets of measured data values. many context prediction approaches utilise knowledge on the frequency of occurence of distinct samples in order to obtain an estimation on the occurence probability of these events. given such probabilities an estimation on the probable continuation of a given time series is possible. This section is designated to provide these basics on probability theory. Some of the examples and illustrative explanations are lend from [74, 62].

5.1 Discussion

Historically, probability theory was disigned to describe phenomena observed in games of chance, that could not be described by classical mathematical theory. A simple example is the description of the outcome of a coin-tossing.

Today we find probability in many aspects of everyday life. An omnipresent example is the weather forecast. What we can learn from it is typically not that it will definitely rain or not, but that a certain probability of rain was derived from distinct measurements at various places all over the world and over a considerable amount of time. Although this information provides no guarantee that it will actually rain or not, we have developed an intuitive understanding of probability so that the information is useful to us although it inherits the chance to be incorrect. Other examples are insurance, where probability is used to calculate the probability of ruin or lottery. The latter example is fascinating as people regularly ignore the infinitisimal probability to win a considerable amount of money.

A further instance where we regularly stumble across probability are quiz shows on TV. Consider, for instance, the following setting (see figure 5.1). A quiz master confronts a candidate with three doors A, B and C and explains that behind one of these the candidate will find (and win) a treasure while the candidate will win nothing if he opens one of the other doors. The candidate then decides for one of these doors, but before it is opened, the quiz master opens one of the remaining two doors and proves that this door does not cover the treasure. The candidate is now given the opportunity to rethink his decision and vote for the closed door he didn't vote for initially.

What should the candidate do in order to maximise the probability to win the treasure? We can show that it is better to alter the initial choice since the remaining closed door containes the treasure with probability $\frac{2}{3}$.

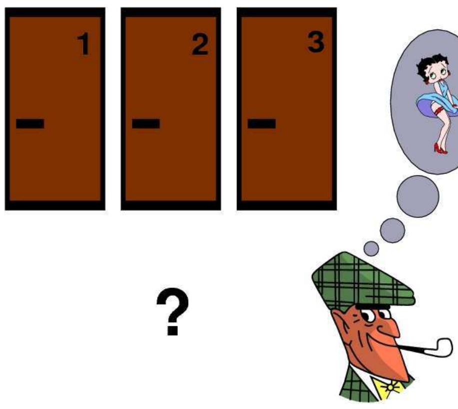


Figure 5.1: Which door hides the treasure?

Exercise 5.1.1 :

Explain why the probability that the treasure is covered behind the remaining closed door is $\frac{2}{3}$.

5.2 Preliminaries

In order to calculate with probabilities we have to define an idealised model. We generally assume that probabilities can be verified by conceptual experiments. This means that the probability for an event represents the quotient of the number of occurences of this event to the number of repetitions of the experiment when the experiment is repreated very often. An event with probability 0.4 should be expected to occur fourty times out of one hundred in the long run. A typical, often quoted example is the tossing of a coin. We expect as an outcome of the experiment one of the two events head or tail with equal probability $\frac{1}{2}$. Note that this assumption is idealised in that we assume a 'fair' coin and disregard some possible but unlikely events like the case that the coin will fall on neither side but, for example, might roll away or stand on its edge.

We have already introduced some typical notation. The results of experiments or observations are called events. Events are sets of sample points. We denote events by capital letters. The fact that a sample point x is contained in event E is denoted by $x \in E$. We write $E_1 = E_2$ when no point $x \in E_1 \wedge x \notin E_2$ or $x \in E_2 \wedge x \notin E_1$ exists. The sample space of an experiment is the set of all posible events. The sample space for the experiment of tossing a coin two times is *head-head, head-tail, tail-head, tail-tail.*

The following examples shall illustrate these concepts

Example 5.2.1 :

Consider the following experiment: Three distinct balls (a,b,c) are to be placed in three distinct bins. The following illustration depicts all possible outcomes of this experiment that together form the sample space

Event	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*
Bin														
1	*abc*			*ab*	*ab*	*c*		*c*		*ac*	*ac*	*b*		*b*
2		*abc*		*c*		*ab*	*ab*		*c*	*b*		*ac*	*ac*	
3			*abc*		*c*		*c*	*ab*	*ab*		*b*		*b*	*ac*

Event	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	
Bin														
1		*bc*	*bc*	*a*		*a*		*a*	*a*	*b*	*b*	*c*	*c*	
2	*b*	*a*		*bc*	*bc*		*a*	*b*	*c*	*a*	*c*	*a*	*b*	
3	*ac*		*a*		*a*	*bc*	*bc*	*c*	*b*	*c*	*a*	*b*	*a*	

However, the sample space is altered when the conditions of the experiment are altered. Suppose that the three balles are not distinguishable. Consequently the following sample space belongs to this experiment.

Event	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Bin										
1	***			**	**	*		*		*
2		***		*		**	**		*	*
3			***		*		*	**	**	*

When we also consider indistinguishable bins, the sample space becomes

Event	*1*	*2*	*3*
Bin			
1	***	**	*
2		*	*
3			*

5.3 Relation between events

We assume an arbitrary but fixed sample space for the remainder of this lecture. The following definitions are essential for our notion of events.

Definition 5.3.1 : Impossible event

With $\chi = \{\}$ we denote the fact that event χ contains no sample points. It is impossible to observe event χ as an outcome of the experiment.

For every event χ there is an event $\neg\chi$ that is defined as 'χ does not occur'.

Definition 5.3.2 : Negation of events

> *The event consisting of all sample points x with $x \notin \chi$ is the complementary event (or negation) of χ and is denoted by $\neg\chi$.*

For two events χ_1 and χ_2 we can also denote new events by the conditions that 'both χ_1 and χ_2 occur' or that 'either χ_1 or χ_2 occur'. These events are denoted by $\chi_1 \cap \chi_2$ and $\chi_1 \cup \chi_2$ resprectively. These events are defined by

$$\chi_1 \cap \chi_2 = \{x | x \in \chi_1 \wedge x \in \chi_2\} \tag{5.1}$$

and

$$\chi_1 \cup \chi_2 = \{x | x \in \chi_1 \vee x \in \chi_2\} \tag{5.2}$$

and can be generalised to arbitrary many events. When the events χ_1 and χ_2 have no sample point x in common, the event $\chi_1 \cap \chi_2$ is impossible: $\chi_1 \cap \chi_2 = \{\}$. We say that the events χ_1 and χ_2 are mutually exclusive.

5.4 Basic definitions and rules

Given a sample space Π and sample points $x_i \in \Pi$, we denote the probability that x_i is observed by $P(x_i)$ with $P : \Pi \to [0..1]$ and $P(x_1) + Px_2 + \cdots = 1$.

Definition 5.4.3 : Probability of events

> *Given a sample space Π and an event $\chi \in \Pi$, the occurence probabiltiy $P(\chi)$ of event χ is the sum probability of all sample points from χ:*
>
> $$P(\chi) = \sum_{x \in \chi} P(x). \tag{5.3}$$

Since all probabilities of the sample space sum up to 1 it follows that

$$0 \leq P(\chi) \leq 1 \tag{5.4}$$

for any event χ.

Consider two arbitrary events χ_1 and χ_2. In order to compute the probability $P(\chi_1 \cup \chi_2)$ that either χ_1 or χ_2 or both occur we add the occurence probabilities that a sample point either in χ_1 or in χ_2 is observed.

$$P(\chi_1 \cup \chi_2) \leq P(\chi_1) + P(\chi_2) \tag{5.5}$$

The '$\leq$'-relation is correct since sample points might be contained in both events. We therefore obtain the exact probability by

$$P(\chi_1 \cup \chi_2) = P(\chi_1) + P(\chi_2) - P(\chi_1 \cap \chi_2). \tag{5.6}$$

Example 5.4.2 :

We toss a coin twice so that the sample space contains the four sample points head-head, head-tail, tail-head and tail-tail that are associated with probability $\frac{1}{4}$ each. Consider the two events χ_1 – head occurs first – and χ_2 – tail occurs second. χ_1 then contains head-head and head-tail while χ_2 contains head-tail and tail-tail. Consequently, $\chi_1 \cup \chi_2$ consists of the sample points head-head, head-tail, tail-tail while $\chi_1 \cap \chi_2$ consists of the single sample point head-tail. We therefore obtain the probability that either χ_1 or χ_2 occurs as

$$P(\chi_1 \cup \chi_2) = \frac{1}{2} + \frac{1}{2} - \frac{1}{4}. \tag{5.7}$$

This can be generalised to higher event counts. For arbitrary events $\chi_1, \chi_2, \dots$ this is expressed by the inequality

$$P(\chi_1 \cup \chi_2 \cup \dots) \leq P(\chi_1) + P(\chi_1) + \dots. \tag{5.8}$$

In the special case that all events $\chi_1, \chi_2, \dots$ are mutually exclusive, we obtain

$$P(\chi_1 \cup \chi_2 \cup \dots) = P(\chi_1) + P(\chi_1) + \dots \tag{5.9}$$

since $P(\chi_i) \cap P(\chi_j) = \{\}$ for any two mutually exclusive events χ_i and χ_j.

In some cases we are interested in the probability that a specific event occurs in the presence of another event. This can be expressed by the conditional probability.

Definition 5.4.4 : Conditional probability

The conditional probability of two events χ_1 and χ_2 with $P(\chi_2) > 0$ is denoted by $P(\chi_1|\chi_2)$ and is calculated by

$$\frac{P(\chi_1 \cap \chi_2)}{P(\chi_2)} \tag{5.10}$$

$P(\chi_1|\chi_2)$ describes the probability that event χ_2 occurs in the presence of event χ_2 (read: The probability of χ_1 given χ_2). With rewriting and some simple algebra we obtain the bayes rule that states

$$P(\chi_1|\chi_2) = \frac{P(\chi_2|\chi_1) \cdot P(\chi_1)}{\sum_i P(\chi_2|\chi_i) \cdot P(\chi_i)}. \tag{5.11}$$

This equation is useful in many statistical applications. Note that the denominator is summed over the probability for all possible events. This means that everything on the right hand side of the equation is conditioned on the events χ_i. When we say that χ_i is the important variable, the shape of the distribution $P(\chi_1|\chi_2)$ depends on the numerator $P(\chi_2|\chi_1) \cdot P(\chi_1)$ with the denumerator as a normalising factor that ensures that the $P(\chi_1|\chi_2)$ sum to 1. The Bayes rule is therefore interpreted that it inverts $P(\chi_1|\chi_2)$ to $P(\chi_2|\chi_1)$. This is useful when it is easy to calculate $P(\chi_2|\chi_1)$ but not to calculate $P(\chi_1|\chi_2)$. With Bayes rule it is then easy to calculate $P(\chi_1|\chi_2)$ provided that we know $P(\chi_2|\chi_1)$ and $P(\chi_1)$.

Definition 5.4.5 : Independence

A collection of events χ_i that form the sample space Π is independent if for all subsets $S \subseteq \Pi$

$$P\left(\bigcap_{\chi_i \in S} \chi_i\right) = \prod_{\chi_i \in S} P(\chi_i). \tag{5.12}$$

Statistical independence is required for many useful results in probability theory. This means, on the other hand, that we have to be careful to apply such results not in cases where independence between sample points is not provided.

Definition 5.4.6 : Expectation

The expectation of an event χ is defined as

$$E[\chi] = \sum_{x \in \mathbb{R}} x \cdot P(\chi = x) \tag{5.13}$$

Although intuitively, the expectation of an event represents the expected outcome of the event, the expectation is not necessary equal to one of the possible sample points.

Consider, for instance, the event χ of throwing a dice. The Sample space is given by $S_\chi = \{1, 2, 3, 4, 5, 6\}$. However, the expectation of this event is

$$E[\chi] = \frac{1}{6} \cdot (1 + 2 + 3 + 4 + 5 + 6) = 3.5. \tag{5.14}$$

It is also possible to perform calculations with expectations of events.

Definition 5.4.7 : Linearity of expectation

For any two random variables χ_1 and χ_2,

$$E[\chi_1 + \chi_2] = E[\chi_1] + E[\chi_2]. \tag{5.15}$$

For an independent random variables χ_1 and χ_2,

$$E[\chi_1 \cdot \chi_2] = E[\chi_1] \cdot E[\chi_2]. \tag{5.16}$$

Definition 5.4.8 : Variance

The variance of a random variable χ is defined as

$$var[\chi] = E[(\chi - E[\chi])^2]. \tag{5.17}$$

For any random variable χ

$$var[\chi] = E[\chi^2] - E[\chi]^2 \tag{5.18}$$

For any independent random variables χ_1 and χ_2

$$var[\chi_1 + \chi_2] = var[\chi_1] + var[\chi_2]. \tag{5.19}$$

For any random variable χ and any $c \in \mathbb{R}$,

$$var[c\chi] = c^2 var[\chi]. \tag{5.20}$$

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.: 1th International Symposium on Handheld and Ubiquitous Computing (HUC99). Volume 1707 of Lecture notes in computer science., 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] Garlan, D., Siewiorek, D., Smailagic, A., Steenkiste, P.: Project aura: Toward distraction-free pervasive computing. IEEE Pervasive computing **4** (2002) 22–31

[8] 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)

[9] 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.

[10] Want, R., Hopper, A., Falcao, V., Gibbons, J.: The active badge location system. In: ACM Transactions on Information Systems. Volume 1. (1992) 91–102

[11] Weiser, M.: The computer for the 21st century. In: Scientific American. Volume 3. (1991) 66–75

[12] Dourish, P.: What we talk about when we talk about context. In: Personal and Ubiquitous Computing. Volume 8. (2004)

[13] Schilit, B.N., Theimer, M.M.: Disseminating active map information to mobile hosts. In: IEEE Network. Volume 5. (1994) 22–32

[14] Schilit, B.N., Adams, N., Want, R.: Context-aware computing applications. In: IEEE Workshop on Mobile Computing Systems and Applications. (1994)

[15] 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

[16] 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

[17] 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

[18] 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)

[19] Dey, A.K.: Providing architectural support for building context–aware applications. PhD thesis, Georgia Institute of Technology (2000)

[20] Mäntyjärvi, J.: Sensor-based context recognition for mobile applications. PhD thesis, VTT Technical Research Centre of Finland (2003)

[21] 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)

[22] 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

[23] 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)

[24] Pascoe, J.: Adding generic contextual capabilities to wearable computers. In: Proceedings of the second International Symposium on Wearable Computers. (1998) 92–99

[25] 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

[26] 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

[27] Chen, G.: Solar: Building A Context Fusion Network for Pervasive Computing. PhD thesis, Hanover, New Hampshire (2004)

[28] Schmidt, A.: Ubiquitous Computing – Computing in Context. PhD thesis, Lancaster University, UK (2002)

[29] Himberg, J.: From insights to innovations: data mining, visualisation, and user interfaces. PhD thesis, Helsinki University of Technology (2004)

[30] 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

[31] 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

[32] Schilit, W.N.: A System Architecture for Context-Aware Mobile Computing. PhD thesis, Columbia University (1995)

[33] 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

[34] 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

[35] 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)

[36] 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

[37] 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

[38] Brooks, R.A.: Elephants don't play chess. In: Robotics and Autonomous Systems. Volume 6. (1990)

[39] 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)

[40] 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

[41] 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)

[42] Rowling, J.: Harry Potter and the Prisoner of Azkaban. Bloomsbury publishing (2000)

[43] 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

[44] 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

[45] 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

[46] 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)

[47] 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

[48] Sigg, S., Haseloff, S., David, K.: Minimising the context prediction error. In: 65th IEEE semi-annual Vehicular Technology Conference (VTC2007-Spring) (to appear). (2007)

[49] 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)

[50] 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

[51] Brockwell, J., Davis, R.: Introduction to Time-Series and Forecasting. Springer (1996)

[52] 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)

[53] 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

[54] Barkhuus, L.: How to define the communication situation: Context measures in present mobile telephony. In: Context, Stanford, CA, Springer (2003)

[55] Greenberg, S.: Context as a dynamic construct. In: Human-Computer Interaction. Volume 16 (2-4)., awrence Erlbaum Associates Inc. (2001) 257–268

[56] Anderson, J.R.: Cognitive psychology and its implications. 3 edn. Spektrum (2001)

[57] 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

[58] 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)

[59] Krsul, I.: Authorship analysis: Identifying the author of a program. Technical report, Department of Computer Sciences, Purdue University (1994)

[60] 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)

[61] Kreiss, J.P., Neuhaus, G.: Einführung in die Zeitreihenanalyse (in German). Springer-Verlag (2006)

[62] Duda, R., Hart, P., Stork, D.: Pattern Classification. 2nd edn. Wiley Interscience (2001)

[63] 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 Bemerkung:.

[64] Wooldridge, M., Jennings, N.R.: Intelligent agents: Theory and practice. In: Knowledge Engineering Review. Volume 10. (1995)

[65] Box, G.E.P., Jenkins, G.M.: Time Series Analysis forecasting and control. Holden-Day (1976)

[66] Wegener, I.: Theoretische Informatik – eine algorithmenorientierte Einführung. Volume 2nd. B.G.Teubner (1999)

[67] Papakyriazis, A., Papakyriazis, P.: Adaptive prediction: a sequential approach to forecasting and estimation of nonstationary environmental systems. In: Kybernetes. Volume 28. (1999)

[68] Laasonen, K., Raento, M., Toivonen, H.: Adaptive on-device location recognition. Number 3001 in LNCS (2004) 287–304

[69] Ashbrook, D., Starner, T.: Learning significant locations and predicting user movement with gps. (2002)

[70] 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)

[71] Ferscha, A.: Pervasive computing. Datenbank-Spektrum (2003) 48–51

[72] Jerome, J.K.: Three men in a boat. Collector's Library (2005)

[73] Sigg, S.: Development of a novel context prediction algorithm and analysis of context prediction schemes. PhD thesis, University of Kassel, Chair for Communication Technology, ComTec (2008)

[74] Feller, W.: An Introduction to Probability Theory and its Applications. Wiley (1968)

[75] Norman, D.: The invisible computer. MIT press (1999)

[76] 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

[77] Gorniak, P., Poole, D.: Predicting future user actions by observing unimodified applications. In: Conference of the American Association for Artificial Intelligence. (2000)

[78] Kolmogorov, A.: Anfangsgründe der theorie der markoffschen ketten mit unendlich vielen möglichen zuständen. In: Mathematiceskii Sbornik. Volume 1. (1936)

[79] Vintan, L., Gellert, A., Petzold, J., Ungerer, T.: Person movement prediction using neural networks. Technical report, Institute of Computer Science, University of Augsburg (2004)

[80] 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)

[81] 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)

[82] 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)

[83] 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)

[84] Silc, J., Robic, B., Ungerer, T.: Processor Architecture - From Dataflow to Superscalar and Beyond. Springer-Verlag (1999)

[85] Pan, S.T., So, K., t. Rahmeh, J.: Improving the accuracy of dynamic branch prediction using branch correlation. In: Proceedings of the fifth international conference on architectural support for programming languages and operating systems (ASPLOS V). (1992) 76–84

[86] Yeh, T.Y., Patt, Y.N.: Alternative implementation of two-level adaptive branch prediction. In: Proceedings of the 19th annual symposium on computer architecture (ISCA-19). (1992) 257–266

[87] Ziv, J., Lempel, A.: A universal algorithm for sequential data compression. IEEE Transactions on Information Theory **23**(3) (1977) 337–343

[88] Petzold, J.: Zustandsprädiktoren zur Kontextvorhersate in Ubiquitären Systemen (in German). PhD thesis, University of Augsburg (2005)

[89] Vapnik, V.N.: Estimation of Dependencies Based on Empirical Data (in Russian). Nauka (1979) (English translation: Springer Verlag, New York, 1982).

[90] Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer (2000)

[91] Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery **2**(2) (1998) 121–167

[92] S.Russell, P.Norvig: Artificial Intelligence A Modern Approach. 2nd edn. Prentice Hall (2003)

[93] 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

[94] Schwaighofer, A., Tresp, V.: The bayesian committee support vector machine. **2130** (2001) 411–417

[95] Kohonen, T.: Self-organised formation of topologically correct feature maps. In: Biological Cybernetics. Number 43 (1982) 59–69

[96] Kohonen, T.: Analysis of a simple self-organising process. In: Biological Cybernetics. Number 43 (1982) 59–69

[97] Kohonen, T.: Self-organisation and Associative Memory. 3rd edn. Springer (1984)

[98] Kohonen, T.: Self-Organising Maps. Volume 30. Springer (1995)

[99] Cottrell, M., Fort, J., Pages, G.: Theoretical aspects of the som algorithm. In: Neurocomputing. Volume 21. (1998) 119–138

[100] 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

[101] 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)

[102] Walter, J., Ritter, H., Schulten, K.: Non-linear prediction with self-organising maps. In: Proceedings of IJCNN. (1990) 589–594

[103] 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

[104] 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

[105] 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

[106] Cottrell, M., Girard, B., Rousset, P.: Forecasting of curves using a kohonen classification. In: Journal of Forecasting. Number 17 (1998) 429–439

[107] Boeckenhauer, H.J., Bongartz, D.: Algorithmische Grundlagen der Bioinformatik. Teubner (2003) (in German).

[108] 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)

[109] 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

[110] 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

[111] 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)

[112] Chatfild, C.: The Analysis of Time Series: An Introduction. Volume 5. Chapman and Hall (1996)

[113] Mehrotra, K., Mohan, C.K., Ranka, S.: Elements of Artificial Neural Networks. MIT Press (1997)

[114] 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

[115] 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

[116] 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)

[117] 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)

[118] Goris, M.J., Gray, D.A., Mareels, I.M.: Reducing the computational load of a kalman filter. In: IEE Electronics Letters. (1997)

[119] 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)

[120] 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

[121] Adams, D.: Mostly harmless. Rel Dey (2000)