Context-aware Querying

Better Answers with Less Effort

Arthur van Bunningen

Samenstelling van de promotiecommissie:

Prof. dr. P.M.G. Apers (promotor)
Prof. dr. L. Feng, Tsinghua University (promotor)
dr. M.M. Fokkinga (co-promotor)
Prof. dr. ir. A.J. Mouthaan (voorzitter en secretaris)
Prof. dr. F.M.G. de Jong
Prof. dr. T.W.C. Huibers
Prof. S. Spaccapietra, Ecole Polytechnique Fédérale Lausanne
dr. ir. A.P. de Vries, TU Delft / CWI Amsterdam

CTIT Ph.D. thesis Series No. 08-115
Centre for Telematics and Information Technology (CTIT)
P.O. Box 217 - 7500 AE Enschede - The Netherlands

SIKS Dissertation Series No. 2008-14
The research reported in this thesis has been carried out
under the auspices of SIKS, the Dutch Research School
for Information and Knowledge Systems.

Cover picture by Zhou Wei (周薇)

*Context-awareness will make using a computer as refreshing as taking
a walk on the Great Wall* (based on Mark Weiser's "Machines that fit
the human environment, instead of forcing humans to enter theirs, will
make using a computer as refreshing as taking a walk in the woods.")

ISBN 978-90-365-2665-4

ISSN 1381-3617 (CTIT Ph.D. thesis Series No. 08-115)

Printed by PrintPartners Ipskamp, Enschede, The Netherlands

Copyright ©2008 Arthur van Bunningen, Enschede, The Netherlands

CONTEXT-AWARE QUERYING

BETTER ANSWERS WITH LESS EFFORT

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit Twente, op gezag van
de rector magnificus prof. dr. W.H.M. Zijm,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen op
vrijdag 13 juni 2008 om 13.15 uur

door

Arthur Hugo van Bunningen

geboren op 25 mei 1979
te Rotterdam

Dit proefschrift is goedgekeurd door:

Prof. dr. P.M.G. Apers (promotor)

Prof. dr. L. Feng (promotor)

dr. M.M. Fokkinga (assistent-promotor)

Preface

Imagine life 20 years from now. Imagine that almost all information about almost everything can be reached within a second. Imagine that you can control almost all devices in your environment with a single interface. What information would you ask for? What action would you take? Wouldn't it be nice if the interface to all this information was aware of your context and provided information tailored to your situation?

Around February 2004, Anton Nijholt pointed me to the database group as a potential place to do interesting research. Coming from Human Computer Interaction, I was a bit suspicious, but I soon learned that the database group had in fact a lot more to offer than SELECT-statements.

First of all, the subject of context-awareness that Peter Apers and Ling Feng offered me is a promise. A promise that, as Mark Weiser suggested, “will make using a computer as refreshing as taking a walk in the woods.” It is also an area that was still in its infancy and these two combined offered so many insights from so many backgrounds, that I sometimes wished that I already had my context-aware interface to these insights. It is an area that triggers your imagination to such extent that you have to force yourself to be realistic in what you can achieve; one of many things I learned about myself these four years.

But more than the subject of context-awareness, the database group offered me an extremely friendly and inspiring research environment. This environment together with all the discussions with and kind words from individuals and groups of people provided the fruitful breeding ground of this thesis. I will name a small subset of them in the following paragraphs.

First of all, this thesis could not have been written without the support of my promotor Peter Apers. I want to thank him for making me feel at home, and finding the time to provide me with valuable feedback and suggestions. Also irreplaceable was my former daily supervisor and now promotor Ling

Feng, whose dedication and kindness have inspired me greatly. But I especially want to thank Maarten Fokkinga, who offered to replace Ling when she moved to China. The time he spent on giving detailed and challenging comments and the hours of discussion were essential to this thesis and to my development the last four years. I am also honored that Stefano Spaccapietra, Franciska de Jong, Theo Huibers and Arjen de Vries agreed to participate in my dissertation committee.

From the database group, I will remember the moments of drinking soup and the accompanying discussions about GO-positions. Especially I want to thank Sandra, Ida, and Suse for all their help and kindness and of course my roommates Vojkan and Henning for being almost like a small family. Henning, also thank you for being one of my paranymfs and for the Friday-sing-a-longs. I am also grateful for the many discussions, especially in groups with abbreviations such as SRO-NICE, AWM, NWO-VIDI (Harold, Sander, Nicolas, Ling and Peter), and HMI; where Dennis deserves a special mentioning for his unique and intelligent view on things, as well as dr. ir. Rutger providing the motivating coffee example on Page 15.

I am also grateful to Professor Lizhu Zhou for having me at the database group of Tsinghua University. And although my time in China was relatively short, it was a unique experience from which I learned a lot. A very special thank you is in place for Li Xiang (李翔); I have never met anyone so unconditionally kind and helpful, and I will never forget how much you know about Dutch computer scientists.

Without friends, nothing is possible and although it is impossible to name every one, I at least want to mention our theatre sport group Ssstille Getuigen, the XXC, and Maarten, Jesper, Stefan, Joost, and Jorien. Jorien, also thank you for being one of my paranymfs and for probably always being the synonym for my pleasant time in Twente.

Last, but not least, I am thankful to my parents and my brother Steven for teaching me to always see the best in someone. And Eric, for the joy and comfort you brought during these exciting years. I am looking forward to experience so much more of our lives together.

Arthur van Bunningen
Enschede, May 2008

Contents

Preface	v
1 Introduction	1
1.1 Introduction	1
1.2 A definition of context	2
1.3 Objectives	3
1.4 Example scenarios	4
1.5 A framework for context-awareness	5
1.5.1 Research questions	7
1.6 Outline of this thesis	7
2 Context data	9
2.1 Characteristics of context data	9
2.1.1 Characteristics of the sources of contextual data	9
2.1.2 Characteristics of the data itself	11
2.2 Implications of using context data	14
2.2.1 User perspective	14
2.2.2 System perspective	16
2.3 The impact on this thesis	18
3 The user's information need	19
3.1 Introduction	19
3.2 Preferences in databases: state of art	20
3.3 Modeling the user's information interest	23
3.3.1 Types of user representation	23
3.3.2 What aspects to represent	25
3.3.3 The Contextual Information Interest Model (CIM)	27
3.4 Representing context and interests in CIM	29

3.4.1	Choice of modeling language	29
3.4.2	Context-aware preference rules in Description logics . .	31
3.5	Addressing uncertainty in CIM	34
3.5.1	Uncertainty in Description Logics	35
3.5.2	Representing uncertainty with events	36
3.5.3	Computing instance membership of concepts	38
3.5.4	Ontological dependencies in probability computation .	40
3.5.5	Vague roles and concepts	46
3.6	Implementation and usage of CIM	47
3.6.1	Implementation of CIM	48
3.6.2	Usage of CIM	50
3.7	Discussion	53
4	Combining preferences	55
4.1	Introduction	55
4.2	State of art	56
4.2.1	Preference as utility	56
4.2.2	Database research in combining preferences	57
4.3	Traditional IR and context-awareness	60
4.3.1	Introduction	60
4.3.2	The query relevance of a document given its features .	61
4.3.3	The effects of context	62
4.4	Combining scores	63
4.4.1	Probabilistic interpretation of the aggregate score . . .	64
4.4.2	Disjunctive context combination	65
4.4.3	Conjunctive context combination	68
4.4.4	Comparison between disjunctive and conjunctive be- havior	71
4.4.5	On the independence of preferences	71
4.5	Score acquisition from historic data	73
4.5.1	Definitions and example history	74
4.5.2	Score acquisition for the conjunctive model	78
4.5.3	Score acquisition for the disjunctive model	78
4.6	Implementing CIM on top of a probabilistic DBMS	80
4.7	Discussion	82
5	Explaining query answers	83
5.1	Introduction	83
5.2	Explaining answers: state of art	84
5.2.1	In database systems	85
5.2.2	Logical explanation of answers	87

5.3	Design Principles	90
5.4	Explaining answers in a probabilistic setting	91
5.4.1	Examples	97
5.4.2	Verifiable views	100
5.4.3	The double fan	101
5.5	Implementing the explanation system	102
5.5.1	Basic Approach	102
5.5.2	Application of existing optimization techniques	105
5.6	Discussion	107
5.6.1	Future research	108
6	Evaluating context-aware querying	111
6.1	Introduction	111
6.2	State of art	112
6.2.1	Evaluation in general	112
6.2.2	Context data for improving query results	113
6.3	What to evaluate?	116
6.4	Assigning the scores by users	117
6.4.1	Ranking methods	118
6.4.2	Measures	119
6.4.3	Results	122
6.5	Score combination functions as classifiers	123
6.5.1	The Naive Bayes classifier	124
6.5.2	Evaluation method	125
6.5.3	On synthetically generated data	126
6.5.4	On real data	128
6.6	Reflections, lessons learnt	129
6.6.1	Future research	130
7	Conclusions	131
7.1	Our achievement	131
7.2	Future research	133
	Bibliography	135
	Samenvatting	151
	SIKS Dissertatiereeks	153

Introduction

From the underlying idea that database systems can support context-aware applications, this thesis explores how context data can help database querying. This chapter introduces the problems that this thesis tries to solve and presents an outline of the rest of the thesis.

1.1 Introduction

Nowadays, more and more information becomes available in digital form. To be able to guide users through this wealth of information, a possibility is to adapt the provided information to the current situation (i.e., context) of the user. In this way the user is able to get better answers from the information with less effort.

Research in this area of context-awareness originates from the vision of Weiser (1991): *to integrate computers in everyday life, . . . , having machines that fit human environment instead of forcing humans to enter theirs*. In our view, for computers to be able to fit human environments, they must be in proper size and shape, appropriate for their users, and adaptable to the user's world; in other words, they should be aware of the user's context.

Suppose you open the door of your car to leave the car and suddenly the car makes an annoying sound. Then you remember that you forgot turning off the lights. You turn them off, and you may ask yourself: was my car context-aware?

Probably it was; the car suspected that you would leave and warned you that you had the lights on, whereas normally there is no sound when the light is on. On the other hand, the car was not so smart to turn the lights off for you. Another, often mentioned, context-aware application is the automatic opening of sliding doors when someone is approaching, or

changing the language of a website based on the language settings of the visiting computer.

Probably you can think of even more examples in which, without explicit interaction, a system based on your situation helps you with something, anticipates you, and guides you. This is the field of applications we are looking at in this thesis.

Nowadays, context-awareness has sparked vigorous discussions in different fields. However, most current context-aware systems and applications are still small-scaled and use limited context data, such as time, location, and user identity. In the data management area, despite some recent attention to the context-awareness issue, the progress is far from the expectation due to the difficulty in capturing, conceptualizing, and representing complicated knowledge about users, context, and tasks (Feng et al., 2004).

1.2 A definition of context

Depending on which school you are from, context-awareness can be placed into the notion of ubiquitous computing, pervasive computing, ambient intelligence, sentient computing, aware computing, invisible computing, wearable computing, etc. They all share to a more or less extent the vision of Weiser.

Since context is such a slippery notion, let us first look at some attempts in the literature to define the notion of *context*. In a broad sense, according to Dourish (2004), “*Context is a slippery notion. Perhaps appropriately, it is a concept that keeps to the periphery, and slips away when one attempts to define it*”. Dourish objects against seeing context as something which is separable from the content of an activity. As an example, he mentions that during a conversation, the location of this conversation could turn from context into content when it becomes the subject of this conversation.

Lieberman and Selker (2000) look at context from a computer programming point of view. Traditionally, the field of computer science tries to be *context-independent*: given the same input, providing the same output independent of the context of the input. They thus come up with a relatively more concrete definition of context. *Context can be considered to be everything that affects the computation except explicit input and output* (Lieberman and Selker, 2000).

Getting close to the application side, one of the most cited definitions of context is from Dey and Abowd (1999):

“Context is any information that can be used to characterize the situation of an entity. An entity can be a person, place, or object that is considered relevant to the interaction between a user

and application, including the user and applications themselves.”
(Dey and Abowd, 1999).

Reverting to the data management field, throughout our study we use the definition of Dey and Abowd, where the interaction is the access of the database by the user.

Usage of context data

According to Dey and Abowd, 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. Dey (2001) makes a distinction between *presentation* of information and services to a user, *automatic execution* of a service for a user, and *tagging information with context* to support later retrieval.

In this thesis, we will not focus on the tagging part but only on the *adaptation of the reaction of the system to the context*, and *proactive context-based system actions*.

Proactiveness, in which the system takes an action without requiring an explicit request of the user, is viewed as one of the most important requirements for a ubiquitous computing environment (Feng et al., 2004). To enable proactiveness, we need effective *information extraction* techniques to identify certain situations and patterns, and some form of *reasoning* mechanisms to determine an appropriate action to take. Tennenhouse (2000) even coins the new term *proactive computing*, which stands for “*the movement from human-centered to human-supervised (or even unsupervised) computing.*”

Since providing proactiveness is a fundamental part of the usage of context data, the word querying in this thesis refers to both traditional pull queries as well as proactive push queries (see also Section 3.6.2).

1.3 Objectives

People claim that “to become credible in the marketplace, context-awareness needs a killer application” (Brown et al., 2000). Typical for killer applications is that it is difficult to predict *whether* a certain application becomes a killer application (e.g., SMS) and *when* a technology will result in a killer application (e.g., hypertext).

Although there is not yet a killer application for context-awareness, there are indicators that context-awareness as a technology is useful as we saw in the typical examples in the introduction. Furthermore, when looking at

personal profile pages on the Internet¹, one can find personal context-aware preferences such as “I like watching romantic films when I’m not in a good mood”, or “I like watching cartoons when I feel sleepless”.

Combining these observations, in this thesis we try to define, research, and evaluate aspects of context-awareness that are largely application independent by providing a general framework.

Proceeding from the high-level objective of having a framework for context-aware applications, in this thesis we consider an approach founded upon database querying.

Through context-aware adaptation of the answers for certain queries to the underlying database system, we achieve context-awareness in an application-independent way.

1.4 Example scenarios

To illustrate the benefits of using context data, let us introduce two examples where the use of context data is beneficial to a user.

First consider the following situation: “*Suzanne is a PhD student, just starting her research in databases. At her faculty they employ a system called AigaionFour to index and annotate the literature. When Suzanne searches for literature, the search results are ranked based on her context; the people she is going to talk with and has talked to, the current paper she is reading or writing, the talks she has just attended, and the people who mailed her. But also based on her ability to work, which for her means in the morning more mathematical papers. Sometimes the system is so sure that a paper is relevant that it will suggest to her the paper even without being queried for it.*”

To illustrate the usage of our framework within new applications we consider, as a second example, the following soft-meeting scenario, “*Peter is a manager at an international consulting company. Most of the time he is with customers. When, however, the opportunity arises, he wants to meet his team members and experts in his area. To support these kinds of meetings, his company employs the soft-meeting planner. With this application, Peter is able to plan so-called soft-meetings; meetings without a fixed time or place. The application notifies Peter whenever the other participants are nearby and available for a meeting. Furthermore, it takes into account the*

¹See for example a Google query for “*I like watching * when*” *site:myspace.com* which returns sentences such as “I like watching TV when I’m not busy working.”

The dialog box 'Plan soft-meeting' contains the following elements:

- Who:** A list box with 'People nearby...' selected. Other options are 'Experts in...' and 'Specific persons...'.
- Subject:** A text field containing '(...)'. Below it is a small icon of a person.
- Method:** A list box with 'Phone' selected. Other options are 'Face2face' and 'IM'.
- Recurrence:** A section with a green arrow icon and the text 'Once a week'.
- Location:** A list box with 'My office' selected. Other options are 'Don't care', 'Meeting room', and 'Availability of...'.
- Buttons:** 'Edit advanced conditions...' and 'OK'.
- Footer:** A small text box containing 'Mood, specific time constraints, etc.'.

Figure 1.1 – Soft-meeting planner: Planning dialog.

The dialog box 'You can have a soft-meeting!' contains the following elements:

- Subject:** XML en Databases
- Location:** ZI3061 [MAP](#)
- People:** You and Sander
- Buttons:** 'Edit meeting...', 'Reject', 'Postpone', 'Ask for approval', and 'Attend unilaterally'.
- Icon:** A small icon of three people in the top right corner.

Figure 1.2 – Soft-meeting planner: Meeting alert dialog.

mood and agenda of Peter himself. At this moment, Peter wants to learn more about XML, therefore he enters a soft-meeting request using the interface in Figure 1.1. When he is in Bucharest and Sander, another English speaking expert in XML, is at the same location, the system is aware of this and notifies Peter that it is possible to have a soft-meeting with Sander (as seen in Figure 1.2)."

1.5 A framework for context-awareness

Since the early years of context-awareness, much research has focused on building architectures (Schilit, 1995), and also more recently there is a focus on designing the right architecture (Costa, 2007). In this thesis, we focus on

Figure 1.3 – A simplified architecture of the system.

a very simple architecture for data(base) driven applications.

In Figure 1.3 we present a simplified view of the system architecture we envision. In our framework, the user (which can be an application) poses a query to an underlying database management system (DBMS). However, in cases where it is appropriate an adaptation layer will, based on a user representation and the current context, present re-ranked query results. What we also envision is that the user is able to manage her own user representation, and that it is possible to (partially) mine the user representation from the interaction history (Log).

To make this architecture more concrete, let us look at the two scenarios. For the literature query, we could imagine that the underlying database management system is something along the lines of CiteSeer (2008) or Google Scholar (2008). The representation contains information such as when, which kind of papers are preferred. When the user poses a query, the results are re-ranked based on the context and representation.

For the soft-meeting planner, when a user requests a soft meeting, the request is stored in the database. The representation contains information about in which situations which kind of meetings the user prefers. The user does a continuous query for beneficial soft-meeting requests and whenever, due to the context, a soft-meeting possibility is considered beneficial, the

system presents the meeting opportunity to the user.

1.5.1 Research questions

Proceeding from the given architecture and focussing on database querying, this thesis intends to answer the following main research question:

- How can context data be used to support database querying?

Given that the context data provides us with information about the information need of a user, we can divide this question into the following three sub-questions:

1. *Which data management topics need to be addressed to incorporate context data into database querying?* Since context data has different characteristics than ordinary data, we review the implications of using this data for database querying in Chapter 2. In Section 2.3, we will further specify this question and discuss the impact on the rest of this thesis. Among others, these topics include the mining of the user representation from the interaction history, addressed in Section 4.5, and the support for traceability of query answers, addressed in Chapter 5.
2. *How can the effect of context data on the information need of the user be stored in a user representation?* Since we want to use context data to improve query results, we need to represent the effect of context data on the results that the user needs. This results in the Contextual Information Interest Model, introduced in Chapter 3 and further elaborated in Chapter 4.
3. *How can the representation of the user together with the context data lead to context-aware query answers?* The application of our model provides the system with a set of documents, scored based on their relevance. In Section 3.6, we discuss how to use this knowledge to re-rank the query results, or proactively provide preferred documents to the user.

1.6 Outline of this thesis

This thesis is organized as follows. Chapter 2 provides an overview of the characteristics of context data and the data management topics that need

to be addressed to incorporate context data into database querying. Section 2.3 elaborates our first sub-question by discussing which of these topics are addressed in this thesis.

Chapter 3 and Chapter 4 introduce our contextual interest model CIM, consisting of scored context-aware preference rules represented in Description Logics, together with a score combination function. Chapter 3 focuses on the choice for preference rules and the representation language. It also shows how the model can deal with uncertain context data using probabilities, and how the model can be implemented on top of an existing relational DBMS. Chapter 4 focuses on the scores of preference rules and the effect of the scores on the combination of preference rules by means of a disjunctive and conjunctive score combination function. The interpretation of the scores and their effects are provided through a probabilistic interpretation of the scores. It also shows how to acquire the scores from an interaction history.

Chapter 5 presents a method for providing explanations for context-aware query answers based on a justification of the ranking of the results. Since we assume that both the scores as well as the context uncertainty can be expressed using probabilities, the chapter is based on an explanation model for probabilistic databases.

Chapter 6 discusses how to evaluate our approach of context-aware querying. It discusses why evaluating complete context-aware applications is hard and presents two preliminary experiments; on the manual assignment of scores for preference rules by users and on acquiring preference scores from an interaction history.

Conclusions and suggestions for future research are presented in Chapter 7.

Context data

Since context is a special kind of data, this chapter discusses the characteristics of context data, and the data management topics that need to be addressed to incorporate context data into database querying. It thereby surveys existing approaches. These topics will be the focus of the rest of this thesis.

2.1 Characteristics of context data

In this section, we describe the various characteristics of context data. These characteristics are highly influenced and determined by the way the data is acquired; we will therefore first characterize the type of data sources.

2.1.1 Characteristics of the sources of contextual data

Context data is sensed through sensors or sensor networks.

One fundamental characteristic of context data is that context data is often sensed through sensors or sensor networks (Akyildiz et al., 2002), for example, location or temperature (Henricksen et al., 2002; Gray and Salber, 2001).

Data management solutions in this field focus on seeing the sensor network as a database. Lazaridis et al. (2004) argue that data streaming approaches are infeasible for the amount of data that those sensors deliver and therefore focus on a quality-driven approach where a query writer can indicate the confidence she wants from an answer (e.g., $\pm 1^\circ C$ of the exact answer). Another system for query processing in sensor networks is TinyDB by Madden et al. (2005). They focus on queries that indicate when, where, and how often the data is acquired. Their approach works on sensors which are running a special operating system (TinyOS), and they try to do as much processing (filtering, aggregation) as possible in the network. Bonnet et al.

(2001) choose a similar database approach to deal with sensor networks in their COUGAR system.

Context data is sensed by small and constrained devices.

What is even more challenging is that, most of the time, context is sensed by cheap, small, and (therefore) constrained devices. Possible constraints are the limited computing power of such devices, the difficulties to run applications on such a low level, and their unreliability (Cherniack et al., 2001). Another serious consequence of the sensor qualities is the battery capacity. Satyanarayanan (2001) and (Lazaridis et al., 2004) both address the issue of energy costs and energy management from a database perspective, whereas Chatterjea et al. (2007) provides a distributed and self-organizing scheduling algorithm for energy-efficient data acquisition.

Context data originates from distributed sources.

As an important aspect mentioned among others by Henricksen et al. (2002), Dey et al. (1999), Goslar and Schill (2004), and Satyanarayanan (2001), contextual information may come from various distributed sources. To get desirable information from these distributed sources, Dey (2001) used aggregators to gather context about an entity (e.g., a person). It is also possible to use sensor querying techniques over these sources, such as the one developed by Lazaridis et al. (2004).

Context data comes from mobile objects.

Closely related to the previous characteristic is the mobility of objects from which the context data is acquired (Satyanarayanan, 2001). According to Jones and Brown (2004), mobility is a prime field for context-aware retrieval due to three reasons: Information is now being made available in situations where it was not available before, a mobile user is often in an unfamiliar environment and needs information about that environment, and it is favorable to use context to help select the information which is needed in this new environment. Satyanarayanan (1996) elaborates two techniques to deal with the mobility of the object and the consequences for information access: *adaptation to the current situation* and *caching*.

Connections between context sensors can be dynamic.

Because of the highly constrained sensors and mobile objects, the connection can be lost when a sensor is out of reach or temporarily unavailable, and has

to be re-established when it is available again. In the meantime, a system could cache the data, or might be able to acquire the information from a not-connected sensor via another sensor or combinations of sensors.

DeVaul and Pentland (2000) present a dynamic decentralized resource discovery framework, which uses semantic descriptions to be able to see what kinds of services are available; the various components are registered to a directory registration service when they are available, and de-registered when they are not available anymore.

Here, it is worth pointing out that the dynamic feature of connections influences the underlying data management strategies. For example, Deshpande et al. (2004) build a statistical model of the sensor data to optimize queries via approximations. Their assumption is that the network topology changes only slowly. Therefore, their techniques are not directly applicable to ubiquitous data management.

2.1.2 Characteristics of the data itself

There are many possible ways to categorize context data (Dey and Abowd, 1999; Chen and Kotz, 2000; Feng et al., 2004; Henriksen and Indulska, 2004). Here, we describe two kinds of categorization methods, namely, *operational categorization* and *conceptual categorization*. Based on how context is acquired Henriksen and Indulska (2004) categorize context into *sensed*, *static*, *profiled*, or *derived* context. Since this categorization is much related to the way context data is acquired, modeled, and treated, we call it **operational categorization**. Contexts of different types differ substantially in how dynamic and reliable they are. In this thesis, we will also refer to the *derived context* as *high-level context*, and to the rest as *low-level context*.

Feng et al. (2004) introduce another context categorization, which distinguishes *user-centric context* from *environmental context* at a conceptual level (See Figure 2.1). We thus call it **conceptual categorization**.

Most of the context categorizations in the literature fall into either of the two kinds (Dey and Abowd, 1999; Chen and Kotz, 2000). These categorizations give us more understanding about which characteristics are applicable to the context data under consideration.

Context is continuously changing.

A crucial property of many sorts of context data is the continuity, i.e., the user's context constantly changes. This may trigger a system to do new actions, resulting in *proactiveness* (Jones and Brown, 2004), but it will also lead to an enormous amount of data to be stored, compressed, and discretized,

Figure 2.1 – Context categorization and acquisition (Feng et al., 2004).

resulting in impreciseness in the database. From a database perspective, the challenges of dealing with continuous data are addressed within so-called *data stream systems* (Babcock et al., 2002).

Context has a temporal and spatial character.

Because of the mobility, both temporal and special data is very important. Koile et al. (2003) propose the notion of “activity zones”, i.e., regions in which the same activities occur, to trigger certain events. Harter et al. (1999) describe a context-aware application which focuses on the user's location using *Bats*, an ultrasound position determination system. Chen et al. (2003) give examples for reasoning with time in temporal ontologies for context-aware scenarios and introduce an ontology for spatial data.

Context data is imperfect and uncertain.

Due to the dynamics of connections, constraints on devices, distribution of sources, and continuity of data, there is a high chance that the acquired context data is not perfect. Henriksen and Indulska (2004) distinguish four types of imperfectness of context data: unknown, ambiguous, imprecise, and erroneous.

Imperfectness can lead to fuzzy situations where it is for example unclear in which room a person is. Both Gu et al. (2004) and Ranganathan et al. (2004) model uncertainty by a probability predicate, and use Bayesian networks for reasoning about dependencies between context elements. The main difference from our approach to uncertainty (discussed in the next chapter) is that they do not focus on querying but only on calculating the probability of high-level context events. Furthermore, these papers present high-level descriptions instead of exact implementations of how to use Bayesian networks for uncertain context data.

Dey et al. (2000) describe a simple architecture for incorporating imperfectly sensed context. Korpipää et al. (2003b) mention uncertainty as well but combine it with fuzzy situation descriptions, for example *Cold*, *Normal*, or *Hot*, and the chances that a situation is like this.

According to the research by Antifakos et al. (2004), displaying an indication of the amount of imperfectness of information to a user will lead to better decision-making.

Next to uncertainty about the current context, we are even less sure about the upcoming context (Cherniack et al., 2001). However, we can try to predict behavior by looking at logged patterns. Furthermore, in some cases uncertainty may even be desirable in order to provide privacy for individuals. This uncertainty links to the quality of available context, and negatively influences context-aware database solutions.

Context data is tightly interrelated.

Not only does high-level (inferred) context data depend on low-level (sensed) context data, but also different kinds of low-level context parameters are inter-related. For instance, the amount of computers in a room and the energy usage of this room are closely related.

This tight inter-relation makes it possible to predict some context parameters based on others (Henricksen et al., 2002). Deshpande et al. (2004) exploit such inter-relations to do optimizations over TinyDB by using correlation between voltage and temperature. However, as noted by Goslar and Schill (2004), because contextual data structures are so highly interconnected, we have to ensure that they are not too complex for limited capabilities of human users and/or local devices. To solve this problem, Goslar and Schill suggest breaking the data structures down into smaller comprehensible parts.

Context data is often personal.

If context data characterizes the situation of a user, this information is personal to this user. This implies a focus on security and privacy for dealing with context data.

2.2 Implications of using context data

Based on the characteristics of contextual data given in the previous section and the usage of context data in Section 1.2, we can identify some data management topics that need to be addressed to incorporate context data into database querying. From the user perspective, there are the demands that users have for applications that use this data. From the system perspective, there are requirements that follow from the kind of data and from the kind of usage of the data. A framework for context-aware applications should deal with both the user perspective as well as the system perspective.

2.2.1 User perspective

For the user perspective, we focus our discussion on non-functional software requirements and arising application possibilities. Beyond the so-called “ilities” (non-functional requirements like reliability, availability, maintainability, responsiveness, manageability, and scalability, etc. (Cherniack et al., 2001)), we identify two major implications on context-aware systems from a user’s point of view that are specific for context-awareness.

Privacy and Security

Because context data is personal, an often mentioned concern for context-awareness is the user’s privacy. For example, early work on context-awareness done by Newman et al. (1991) evidenced that during experiments with tracing users during the day with badges, users did not wear them because of privacy issues.

However, Kindberg et al. (2004) draw the conclusion that other aspects, such as usability, are at least equally important to users. Furthermore, Kindberg et al. find that using visible tangible objects to do transactions (e.g., a barcode scanner) can make transactions more trusted by the users.

Gandon and Sadeh (2004) present in their research an interesting solution to deal with the privacy and security issue. The context data of users is stored in a so-called *e-Wallet* and it uses both access rules and obfuscation rules to deliver different context data to different users or applications. Leonhardt

and Magee (1998) report on access rules with a focus on the user's location context. Finally, van Heerde et al. (2007) address privacy in databases, balancing potential use of the data and privacy breach using degradation policies.

Traceability and correction

Since context data is uncertain, some decisions that a system makes, based on this information, might not be correct. Furthermore, because context changes may lead to different system reactions to the same user action, the user might want to know why something (proactively) happened.

Ideally we would like proactiveness to be understandable and controllable by users, as suggested by DeVaul and Pentland (2000). As Dey et al. (1999) put it, "*we would ideally handle context in the same way as user input.*"

Chalmers (2004) mentions that it should be possible to focus on the tool (the computer) to have it "present-at-hand" in ubiquitous scenarios. This results in the following three system design principles:

- Systems should display their own internal states and configuration to the users.
- The deep system structure should be revealed so as to support inspection and adaptation.
- Interfaces should offer "direct experience of the structures by which information is organized."

Here, two remarks should be made. First, for some information it suffices not to be visible at all times but only on request, besides, in many cases it is sufficient or even better to give a conceptualization or abstraction of the internal state. An example here could be the dashboard of a car, by which the user can have the car present-at-hand in case something goes wrong. Another example is the network signal indicator of a mobile phone (Chalmers, 2004).

Next to having the possibility to focus on the tool, the user should be able to use this information to correct or improve the system. For example, suppose a context-aware coffee machine learns that the user always makes coffee at workdays at seven o'clock. The coffee machine therefore decides to proactively make coffee at seven. Introspection by the user makes it possible to improve the system and make it prepare the coffee at ten minutes before seven.

Possibility to correct information can also be seen as *mediation*; a dialogue between the user and the system. Dey and Mankoff (2005) discuss mediation in context-aware systems from an architectural point of view.

2.2.2 System perspective

From the standpoint of systems, context-awareness raises a number of challenges to ubiquitous data management.

Support for data and sensor characteristics

First of all, a system could provide native support for the previously identified characteristics of the contextual data. For example, it makes sense to support spatial queries or have a database that supports uncertainty, although uncertainty is just one type of meta-data that is important for context data. Gray and Salber (2001) distinguish the following categories of meta-data: forms of representation, information quality, sensor source, interpretation (data transformation), and actuation (for example, to shut down faulty sensors).

Also, a context-aware system might support the distributiveness and dynamic connections through a distributed architecture or using a goal-oriented approach for acquiring the data.

Learning and reasoning

Since the context is sensed through small and constrained devices, most of them are only able to sense low-level context. Therefore a contextual framework should support the inference from low-level to high level-context and interpret it in terms that accurately reflect human perception of tasks and needs. Schmidt (1999) is one of the first who did so by using *cues*, which take the value of one sensor and provide a symbolic or sub-symbolic output. Taking the output “*the user is running*” and “*the user has a high pulse*” for example, according to several of those cues, a context such as “*the user is jogging*” can be determined. Korpipää et al. (2003a) exploit a set of techniques including Bayesian networks to recognize high-level context, as seen in their procedure in Figure 2.2.

Doing reasoning calls for a way to represent knowledge. Strang (2004) gives an overview of representation languages for context. A more detailed discussion of modeling context is given in the next chapter.

To get more knowledge from sensor data using logic programming, Whitehouse et al. (2005) develop the concept of semantic streams, in which services transform low-level sensor data into more meaningful data.

Furthermore, to support proactiveness and adaptation while being unobtrusive, a context-aware system should be able to automatically learn the behavior of the user and her preferences.

Figure 2.2 – Context derivation layers from Korpipää et al. (2003a)

Schema and data integration

Confronted with different context data from diverse sensors and possibly from different domains, a context-aware system needs a flexible context representation mechanism so as to provide conversion among different kinds of context data. Bressan et al. (1997) discuss a method of using Prolog rules to convert between different representations. This is similar to the schema and data integration problem, which has been extensively addressed by Levy (2000) using Description Logics (Baader and Nutt, 2003). Although we will not address this topic in the rest of this thesis, it might be addressed orthogonally, possibly resulting in more data uncertainty de Keijzer (2008).

Storage and logging of context data

Because context-awareness implies to be proactive and to detect patterns in the user's behavior, context data and related reactions thus need to be stored somewhere (ECHISE, 2005). Hereby, a number of questions related to what, where, and how to store context data arise (Meyers and Kern, 2000).

2.3 The impact on this thesis

This chapter answered the first part of the first research question about the data management topics that need to be addressed to incorporate context data into database querying. Given the observations in this chapter, we can elaborate on the second part of the question, namely, how to address these topics.

Since it is impossible to address all topics in one thesis, we will focus on a subset that presents a working starting point. In practice, this means that we will not address the integration of context data. In addition, we will not explicitly address privacy and security. However, the architecture that we envision (see Section 1.5 and Section 3.6.2) supports the storage of the user representation and personal data on a different server from the application data, which at least provides a basis for user control over their data. Finally, we focus on uncertainty as data characteristic because of its relation to the scoring of preference rules. This leads to the following elaboration of our first research question:

1. *How to address traceability and correction?* Traceability is addressed in the representation language of our model (Section 3.4.1) and in the score combination model (Chapter 4). Furthermore, we discuss how to provide traceable query answers through explanation (Chapter 5).
2. *How to address data and sensor characteristics (uncertainty)?* From the data-characteristics, we focus on uncertainty in Section 3.5.
3. *How to address learning and reasoning?* Reasoning forms the basis of the choice for our representation language in Section 3.4.1, whereas we will show how to learn preferences in Section 4.5.
4. *How to address storage and logging of context data?* To be able to easily store and log context data, we discuss in Section 4.6 how our model can be implemented on top of a DBMS.

The user's information need

This chapter introduces the contextual information interest model CIM, aimed at modeling the context-aware information need of the user. The chapter focuses on the choice for preference rules as basic building blocks for the model and their representation language: Description Logics. It also shows how to address uncertainty of context data using probabilities and how to implement the model upon an existing relational database.

3.1 Introduction

Suppose that you are at a cocktail party. You see a woman holding a martini glass, which you believe to contain water. However, you believe that everyone else believes (and believes that you believe) that the glass contains martini. When you are talking to your colleague Smith, you would understand that Smith refers to this woman via question (1) but not via question (2):

- (1) Who is the woman holding the martini?
- (2) Who is the woman holding the water?

The reason is that you do not believe that Smith knows about the water in the glass. Also, if you would like to refer to the woman when talking to Smith, you would do so using question (1), since you would not think that Smith will pick out the intended woman based on question (2).

This example, from research on dialogue systems by Perrault et al. (1978), makes clear that even in normal conversations people (unconsciously) make use of quite complex representations of users to adapt their behavior to the environment.

Similarly, to use context data to adapt or bring about system behavior, a representation of the user is needed that describes the relation between the context and the information need.

This chapter introduces a scheme to describe this representation: the contextual information interest model CIM. The model consists of so-called *preference rules* that each record an information interest of the user in a certain context. Both preference and context are represented in Description Logics, and a score indicates the strength of a preference rule.

As the user is represented by her preferences, we first discuss previous approaches on the use of preferences for database querying in Section 3.2. The second part of this chapter then discusses the basics of our model:

- We discuss possible ways to represent a user and introduce our model in Section 3.3.
- We motivate and discuss the representation language that is based on Description Logics in Section 3.4

The third part of this chapter is concerned with two of the data management topics that need to be addressed to incorporate context data into database querying.

- We discuss how to address uncertainty of context data in our model in Section 3.5. We show how to deal with uncertainty in Description Logics queries through probabilistic event expressions and address the influence of having an ontology on the calculation of the probabilities.
- In Section 3.6 we address the storage and logging of context data by implementing the model on top of a DBMS and show how we use the extra knowledge of the user's information need in query processing.

We end this chapter with a summary of our findings in Section 3.7.

3.2 Preferences in databases: state of art

As we base our model on the preferences of a user and one of the objectives is to use existing database techniques, we will in this section discuss the state of the art in preferences and databases.

Lacroix and Lavency (1987) were the first to introduce the notion of a preference query to the database field. They extended the Domain Relational Calculus to express preferences for tuples satisfying certain logical conditions. Since its introduction, extensive investigation has been conducted, and two main types of approaches have been formed in the literature to deal with the user's preferences, namely, *qualitative* and *quantitative* (Chomicki, 2003).

The qualitative approach intends to directly specify preferences between the tuples in the query answer, typically using binary preference relations.

An example preference relation is “*prefer one book tuple over another if and only if their ISBNs are the same and the price of the first is lower.*” These kinds of preference relations can be embedded into relational query languages through relational operators or special preference constructors, which select from their input the set of the most preferred tuples; this approach is, among others, taken by Chomicki (2003) using the *winnow* operator, Kießling (2002) in their PreferenceSQL *best match only* model, and Börzsönyi et al. (2001) in skyline queries.

To get an idea of the representation for this approach, consider the following preference from Chomicki (2003), which specifies a preference of white wine over red when fish is served, and red wine over white, when meat is served, over a relation *Meal(Dish, DishType, Wine, WineType)*:

$$\begin{aligned} (d, dt, w, wt) \succ (d', dt', w', wt') \equiv & (d = d' \wedge dt = \text{'fish'} \wedge wt = \text{'white'} \\ & \wedge dt' = \text{'fish'} \wedge wt' = \text{'red'}) \\ & \vee (d = d' \wedge dt = \text{'meat'} \wedge wt = \text{'red'} \\ & \wedge dt' = \text{'meat'} \wedge wt' = \text{'white'}) \end{aligned}$$

Another example representation of a qualitative preference, is the following preference for red or blue cars and (equally important) maximal *fuel_economy*, using preference XPATH from Kießling et al. (2001):

```
/CARS/CAR #[ (@color) in ("red","blue")
               and (@fuel_economy) maximal ]#
```

Note that this example already includes composition of preferences through the *and* operator. We will look into this aspect in detail in the next chapter.

The quantitative approach expresses preferences using scoring functions, which associate a numeric score with every tuple of the query. Then tuple t is preferred over tuple t' if and only if the score of t is higher than the score of t' . Agrawal and Wimmers (2000) provided a framework for expressing and combining such kinds of preference functions. Koutrika and Ioannidis (2005) presented a richer preference model which can associate degrees of interest (like scores) with preferences over a database schema. Figure 3.1 gives a graphical representation of the preference specification of Koutrika and Ioannidis, where both attributes and relations receive scores.

Since our model as defined in Section 3.3.3 compares documents based on whether preference rules are applicable to the documents individually using scores, it is similar to the quantitative approach.

Recently, context-aware preferences start to receive attention because user preferences do not always hold in general but may depend on underlying situations (Holland and Kießling, 2004; Stefanidis et al., 2005; Stefanidis and

Figure 3.1 – Personalization graph in the preference model of Koutrika and Ioannidis (2005), indicating among others a preference for tickets around 6 euros and a big concern about the genre of a movie. The first value between brackets indicates the preference for the *presence* of the associated value, the second indicates the preference for the *absence* of the associated value. The function e stands for an elastic preference.

Pitoura, 2008). Holland and Kießling (2004) used the ER model to model situations. Each situation has an *id* and consists of one time stamp and one location. It can also have one or more influences (e.g., a personal and a surrounding influence). They link situations with uniquely identified preferences through an $m : n$ relation and use an XML-based preference repository to store and manage the situated preferences. Stefanidis and Pitoura (2008) take context as any attribute that is not part of the database schema and focus on the hierarchical nature of context attributes, such as streets, cities and countries for a location-attribute. They introduce so-called *contextual preferences* consisting of a context state, a predicate, and a score. The predicate specifies conditions on the values of attributes that hold in the context-state. As an example of a preference in their model, the preference $((id1, youth, male), friends), (genre=thriller), 0.9)$ denotes that the user with *id* *id1*, who is a young male, when accompanied with friends, enjoys to see movies of thriller genre.

Regarding preferences, our work distinguishes from the previous work in two aspects. First, addressing the learning and reasoning implication, we propose a knowledge-based context-aware preference model, where both contexts and preferences are treated in a uniform way using Description Logics as introduced in Section 3.4. Second, we show how our model is able to deal with other implications of context-awareness such as uncertainty (see Section 3.5). Other major differences are given in the following chapters where

we focus on probabilistic interpretation of the score functions in Section 4.4.1 and explainability of the model in Chapter 5.

3.3 Modeling the user's information interest

This section builds on the work from McTear (1993) and Rich (1983) to discuss possible ways to represent users and introduces our model.

We start by discussing the possible types of representations, which types are typical for context-awareness and which should be supported by our model. After that we discuss possible aspects of the user that can be used as a representation and argue that our model should be based on preferences. Finally, we present in Section 3.3 our representation model consisting of preference rules.

We will consistently use the term *model* for the schema in which the users can be represented; thus a user *representation* is an instance in the model and it describes a single user. Furthermore, we will use the word *system* for the complete user modeling system.

3.3.1 Types of user representation

In the work of McTear and Rich, the following four characteristics of user representation types are identified:

Canonical versus individual Rich was one of the first who made the distinction between canonical and individual representations. Whereas a canonical user representation represents “the average user,” an individual user representation represents a single user. In some systems, a middle course between canonical and individual representations is chosen by representing groups of users; the so-called *stereotypes* as introduced in the Grundy system (Rich, 1998).

Static versus dynamic This characteristic is concerned with the modifiability of the representation. A representation can be either modifiable (dynamic) or non-modifiable (static). A system that uses dynamic representations is able to adapt them as the characteristics of the user change. Static representations might contain user properties like gender and date of birth, since these properties do not change over time.

Long-term versus short-term The first representations of users were concerned with dialogues. In these representations there was a clear distinction between user knowledge that was kept, and the knowledge that

was discarded at the end of a dialogue (session). But also in other tasks where representations of users are currently used, one can identify sessions, for example a search task within a retrieval system. Having a short-term representation for specific knowledge can be useful to limit the time it takes to infer user properties. For example, within the retrieval system, such a representation might consist of the knowledge of the information-seeking task that the user is currently undertaking. Long-term representations are more usable for general knowledge about the user.

Implicit versus explicit acquisition The representation of the user can be acquired through explicit user input or through interaction of the user with the system.

Context-aware versus context-independent A new characteristic of the representation of users that we like to identify is the distinction between context-aware versus context-independent representations. Traditionally this distinction was not made and whether the representation was context-aware was often implicit, although there were examples of context-aware representations in early literature; e.g., the starting example of this chapter. Our motivation for this distinction is that the characteristic of context-awareness may induce other characteristics and requirements like the ones discussed in the previous chapter. Among others, it introduces the property that the same user might have a different representation in different situations.

Although we can use these characterizations to distinguish representation types, a context-aware representation should be able to have different representation types for different properties.

Especially, context-aware representations should support *both canonical and individual* properties, where in the former case, the representation might contain some general knowledge such as “the *average user* prefers to have an umbrella when it rains, but sunglasses when the sun shines.” Furthermore, the context-aware representation that we envision (see the requirements in the previous chapter) is *inherently dynamic*, for example, to support correction of the stored knowledge. Context-aware representations have *nearly always a long-term* character, since they express the difference in behavior over different situations and are therefore only useful if the session contains multiple situations. Since one of the goals of context-awareness is that it will lower the effort of the user, a context-aware representation will *mostly be implicitly acquired* through interaction, however, it should support explicit modification of the properties for bootstrapping or correction. Finally,

Figure 3.2 – The Microsoft office assistant: based on a representation of the user, using goals and needs.

a context-aware representation will contain *both context-aware and context-independent* properties.

Since our model is used to describe a context-aware representation, it should support these representation types. This implies the support for canonical and individual properties given in Section 3.3.3, the modification of the model through a preference manager in Section 3.6.2, the support for acquisition of the representation from the history as described in Section 4.5, and the support for context-aware and independent properties as given in Section 3.4.2.

3.3.2 What aspects to represent

In general, a representation of a user can be used to describe the (expected) behavior of the user to get either more insight in the behavior of the user, or to be able to give a better system response to the user.

Such representation can be either a deep psychological representation or a simpler one, depending on the usage. Based on the work of McTear (1993) we identify four sets of possible aspects to represent and show how these pieces of information could help a context-aware ticket vending machine.

Goals and plans (intentions) The first aspect of the user that can be used to represent the user are her goals and plans. These plans can be used to interpret the interaction of the user with the system. For example, suppose that a user at Rotterdam central station presses the

button 'Utrecht' on the vending machine. If the system knows that the goal of the user is to depart, it can present the user with possibilities for leaving Rotterdam to Utrecht. If the system knows that the goal of the user is to pick someone off the train, it can show the arrival times of trains coming from Utrecht. An exemplary example of a system that represents the user by her goals is the Lumière project by Horvitz et al. (1998), that led to the Microsoft office assistant (see Figure 3.2). In this project, Horvitz et al. considered next to the goals also the *needs*; pieces of information or actions that would reduce the time or effort to achieve the goals.

Capabilities The second aspect is the capabilities of the user. For example, her expertise or the capabilities of her devices, such as the screen resolution. In our ticket-vending example, when the ticket-vending machine would know that the user is blind, it could, instead of visual feedback, provide either tactile or audio feedback.

Beliefs and knowledge Related to the capabilities of a user is her knowledge. An example of this aspect was the knowledge about the belief of others that was necessary to refer to the woman with the water, in the introductory example. In the ticket-vending example, the ticket-vending machine could present the user with a route to the platform if the user does not know how to get there (or the user's beliefs about the location are wrong, because of construction work).

Preferences To predict how the user evaluates system choices, one can represent the preferences of a user. One of the first systems that represented users by their preferences, was Grundy (Rich, 1998), which used preferences for books. In our example, the ticket machine could be based on a preference of the user for first class seats, first show the first class options for traveling.

Recall that the central question in this thesis is to determine what information is needed by the user that is not explicitly expressed in the query. Since information need could be seen as missing knowledge of the user, one might be tempted to represent the user by her beliefs or knowledge. However, since the amount of possible user knowledge is extremely large, we strive for a simpler representation by assuming that the information need of a user can be represented by *preferences* for pieces of information. The advantage of such a representation is that it directly relates to the user evaluation of a result; a higher preference for a result means that the user rates such a result higher. We might conclude from this decision that the purpose of our final

Figure 3.3 – The rule designer from Dey et al. (2006): The rule constructed is “If the phone rings in the kitchen and music is on in an adjacent room, or if the phone rings in the kitchen and Joe is outside, then turn up the phone volume in the kitchen.”

system will actually be to present the user with answers that best serve the *information preference*, instead of information need.

3.3.3 The Contextual Information Interest Model (CIM)

As our goal is to model the *information preference* of the user while taking into account the context, we need to represent the preference relative to the context. An often used technique is to use so-called “IF-THEN rules”, which were used in one of the first context-aware applications; the ParcTab environment (Schilit et al., 1994). Schilit et al. even described context-awareness as “Something like living in a rule-based expert system.” Furthermore, Dey et al. (2006) gave support that people express their own context-aware preferences mostly using “If I ...” or “When I ...” forms. Their model results in a rule designer based on a situation-action interface as seen in Figure 3.3. Based on this evidence and because of the simplicity of the approach, that can facilitate traceability, we model the *preferences* of a user using a *rule based* approach.

To introduce the model, we first postulate the existence of the set of all possible situations Sit , and the set of all possible documents Doc . A situation $s \in Sit$ is equal to a single possible situation and a document $d \in Doc$ is

equal to a single document. In the scope of this study the granularity of information is “document” as is typical in the information retrieval field. In the database field one can substitute document with “record”.

Definition 3.3.1. A context-aware preference rule is a tuple of the form $(\mathcal{C}, \mathcal{I})$, where $\mathcal{C} \subseteq \text{Sit}$ is called the context, and $\mathcal{I} \subseteq \text{Doc}$ is called the information interest.

The intended interpretation is that a situation s in $\mathcal{C}$ leads to an interest in documents in $\mathcal{I}$. In this thesis we will often use *preference rule* as shorthand for context-aware preference rule. Both context and information interest are represented in Description Logics, which is detailed in Section 3.4.2.

To support our discussion on preference rules, we define the concepts of applicability for preference rules, and satisfiability of interests and preference rules.

Definition 3.3.2. Let $r = (\mathcal{C}, \mathcal{I})$ be a preference rule, and d and s be a document and a situation, respectively. Then r is applicable in s if $s \in \mathcal{C}$.

Definition 3.3.3. A document d satisfies an interest $\mathcal{I}$ if $d \in \mathcal{I}$.

Definition 3.3.4. A document d satisfies a preference rule $r = (\mathcal{C}, \mathcal{I})$ in a situation s , denoted as $r \models_s d$, if r is applicable in s and d satisfies $\mathcal{I}$.

For each user u we postulate a set rs_u of preference rules. The intended interpretation is that this set of preferences expresses the query independent (or *implicit*) information preference for a user. Together these preferences form the representation of the user. A preference r is *canonical* if $r \in \bigcap_u rs_u$.

To indicate the strength of the users preference in case a document is satisfied by multiple preference rules at the same time we introduce a score function σ , which given a context-aware preference rule, results in a score. The semantics, and therefore the effect on rule combination, are discussed in the next chapter.

In the rest of this chapter we are concerned in situations in which the user has only a single preference rule r . In this case, the preference of a document d_1 over a document d_0 in situation s ($d_0 \prec_s d_1$) is a binary decision defined in terms of the satisfiability of r :

$$d_0 \prec_s d_1 \Leftrightarrow (r \not\models_s d_0) \wedge (r \models_s d_1) \quad (3.1)$$

We shall in Section 3.5.5 see that, if we pose no restriction on the preference rules, one rule is as expressive as multiple rules.

3.4 Representing context and interests in CIM

To express a context-aware preference, we need to have a representation for context and interests. In this section we will show various requirements for such a representation. We will discuss existing representation methods in the area of context-awareness and present our representation based on Description Logics, addressing some of the specific demands of context-aware systems.

3.4.1 Choice of modeling language

In the literature, there exist several possibilities to model context. Most of them are surveyed by Strang (2004), who identified six major types of context modeling. For other types of representation, please refer to the work of McTear (1993).

Key-value models In key-value models, context data is provided as a value of an attribute, e.g., ‘*Location = ROOM3061*’.

Markup scheme models Markup scheme models consist of a hierarchical structure with attributes and content. For example, the W3C standard on Composite Capability/Preference Profiles.

Graphical models Models such as object role modeling, and the unified modeling language.

Object-oriented models A more programmatic-oriented approach, where implementation/calculation details are hidden in objects behind interfaces.

Logic-based models The classical formal models of context based on the work of McCarthy and Buvac (1997).

Ontology-based models Models based on variants of the Web Ontology Language (OWL).

Based on existing implementations of the approaches, Strang concluded that *ontology-based languages* are preferable for context modeling, mainly because of their possibility for distributed knowledge composition, the possibility for partial validation of the knowledge, and their level of formality.

Furthermore, as indicated in Section 2.3 the preference model must possess reasoning and learning capabilities, calling for an inevitable knowledge ingredient. For example, a user may input a preference like *prefer a nearby*

restaurant when the weather is bad. With the model, it should be able to infer the applicability of the preference no matter whether it rains or snows, since both are bad weather. Also, one of the important data management topics that we identified was traceability (e.g., the preference model should be traceable by the users). In other words, it should be possible for a human to conveniently enter, view, and edit context-aware preferences in a way which is close to the world model of the users. Both implications can be addressed in ontology-based languages.

Motivated by the results of Strang and the fact that existing database approaches, as surveyed in the previous section, do not address the combination of reasoning and traceability requirements, we exploit a variant of Description Logics (DL) for representing both the contexts and interests of the users.

Description Logics (Baader et al., 2003a) is a family of knowledge representation languages that are (most of the time decidable) fragments of first order logic. They form the basis of ontological languages such as OWL, which has been used to model context by Chen et al. (2005) and Wang et al. (2004). This translation to OWL also makes it possible to share information between separate ontologies, which allows one to use existing ontologies such as OWL-S, Friend-Of-A-Friend, DAML-TIME etc. (Preuveneers et al., 2004; Chen et al., 2005). Furthermore, there exist many tools for dealing with Description Logics knowledge bases such as reasoners (e.g., Pellet and Racer (2008)) and editors, (e.g., Protégé (2008)), as seen in Figure 3.4) supporting the reasoning and traceability requirements. Finally, extensive research has been conducted to investigate the relationship between databases and Description Logics, and to map a Description Logics knowledge base into database schemas (Borgida, 1995).

Remarks

Of course it should be noted that it is possible to convert other formal or logic-based approaches, such as conceptual graphs or OpenCyc, to Description Logics. Therefore, our Description Logics based model should be seen more as an example with useful properties, than as a strict requirement for modeling the context-aware preferences of a user. Interestingly, the ancestors of Description Logics such as CB-ONE and KL-ONE were once used to model users in terms of beliefs (Kobsa and Pohl, 1994).

Rest to say, that, although the relation of Description Logics to OWL makes it easy to define our own context-aware ontology or to choose between existing ontologies, the purpose of this thesis is not to decide on the best ontology. Rather we look into some constructors that the ontology language

Figure 3.4 – Screenshot of editing a Description Logics concept in Protégé.

should deal with, such as uncertainty.

3.4.2 Context-aware preference rules in Description logics

A Description Logics knowledge base consists of both domain knowledge (i.e., the vocabulary/schema) and instance knowledge. The domain knowledge contains (intensional) assertions about *concepts* (e.g., Child, Female) and *roles* (e.g., hasRoom, hasActivity). The instance knowledge contains (extensional) assertions about *instances* (e.g., ROOM3061). Concepts and roles are either *atomic* or *constructed* using concept and role constructors *intersection* ($\sqcap$), *union* ($\sqcup$), and *complement* ($\neg$) (e.g., Child $\sqcap$ Female, hasRoom $\sqcap$ hasActivity). A concept specific constructor is the *one-of* construct ($\{a_1, \dots, a_n\}$) which defines a concept consisting of a specific set of instances $\{a_1, \dots, a_n\}$. The *top concept* ($\top$) and *bottom concept* ($\perp$) denote respectively all instances and no instances. A role specific operator is the *role-inverse* which defines the inverse of a certain role (e.g., roomOf is the inverse of hasRoom, denoted as $hasRoom \equiv roomOf^{-1}$). Moreover, roles can have universal and existential quantification (e.g., $\forall hasChild.Female$ denotes the set of instances whose children are all female, and $\exists hasChild.Female$ denotes the set of instances having a female child). A specific case of existential quantification is the *exact number restriction* ($\exists_n$) for roles (e.g., $\exists_1 hasRoom.Room$ denotes the set of instances who are in exactly one room). The basic inference on concepts in Description Logics is *subsumption* $C \sqsubseteq D$, which is

$Person \sqsubseteq Thing$	$ActType \sqsubseteq Thing$
$\sqcap \forall hasRoom.Room$	$FreeTimeAct \sqsubseteq ActType$
$\sqcap \forall hasActType.ActType$	$ChatActivity \sqsubseteq FreeTimeAct$
$\sqcap \forall hasColleague.Person$	$EatActivity \sqsubseteq FreeTimeAct$
$\sqcap \exists_1 hasRoom.Room$	$Location \sqsubseteq Thing$
$Document \sqsubseteq Thing$	$Topic \sqsubseteq Thing$
$\sqcap \exists hasTopic.Topic$	$hasAuthor \equiv authorOf^{-1}$
$\sqcap \forall hasAuthor.Person$	$hasRoom \equiv roomOf^{-1}$
$Room \sqsubseteq Location$	

Figure 3.5 – An ontology example using Description Logics.

to check whether the concept denoted by D (the *subsumer*) is more general than the one denoted by C (the *subsumee*). Furthermore an instance a can be a *member* of a concept C , denoted $C(a)$, and similarly, a tuple of two instances can be a member of a role, denoted $R(a, b)$. Baader et al. (2003a) discuss more details about Description Logics and give an overview of other possible constructors.

We use Description Logics to describe a world model (i.e., ontology) upon which our context-aware query preferences are founded. An ontology example is given in Figure 3.5, where concepts are written in CamelCase (e.g., *OfficeActivity*), roles in lowerCamelCase (e.g., *hasRoom*), and instances in all capital letters (e.g., *ROOM3061*). Finally we assume that each object has a unique name and (hence) distinct instance names denote distinct objects (*unique naming assumption*).

We express various contexts of query preferences via Description Logics concepts. For example, consider the following concept:

$$\{PETER\} \sqcap (\exists hasActType.FreeTimeAct) \\ \sqcap (\exists hasFriend.(\exists hasRoom.(\exists roomOf.\{PETER\})))$$

It describes a situation in which user *PETER* is doing some free time activity with at least one friend in the same room.

Beyond contexts, Description Logics concepts also can be used to convey information needs. For instance, the concept

$$Document \sqcap (\exists hasTopic.\{XML\})$$

can be viewed as a query which selects all *Document* instances with the *XML* topic. Therefore, in a similar fashion as situations, we describe the user's preferences through Description Logics concepts.

Throughout the study, we stipulate that all knowledge about the current situation is stored as instance knowledge. Similarly, we consider that all (identifiers of) documents with their properties are stored as instances in the knowledge base.

As a result, the set $\mathcal{C}$ contains situations in which the Description Logics representation of the context has at least one instance as a member and the set $\mathcal{I}$ contains documents that are members of the Description Logics representation of $\mathcal{I}$.

As an example, user *PETER*'s context-aware query preference “*prefer documents about the news while doing some free time activity*” can be specified as $(\mathcal{C}, \mathcal{I})$ where

$$\begin{aligned}\mathcal{C} &= \{PETER\} \sqcap (\exists hasActType.FreeTimeAct) \\ \mathcal{I} &= Document \sqcap (\exists hasTopic.\{NEWS\})\end{aligned}$$

This $\mathcal{C}$ contains all those situations in which Peter is doing a free time activity:

$$\begin{aligned}\mathcal{C} = \{ &\{\dots, hasActType(PETER, LUNCH), \\ &\quad hasRoom(PETER, ROOM3061), \dots\}, \\ &\{\dots, hasActType(PETER, SPORTING), \dots\}, \\ &\{\dots, hasActType(PETER, LUNCH), \\ &\quad hasRoom(PETER, ROOM4061), \dots\}, \\ &\dots\}\end{aligned}$$

Similarly, $\mathcal{I}$ contains all those instances that are documents about news. If these documents can be uniquely identified with their attributes, $\mathcal{I}$ looks as follows:

$$\begin{aligned}\mathcal{I} = \{ &\{\dots, hasTopic.\{NEWS\}, hasLength.\{SHORT\}, \dots\}, \\ &\{\dots, hasTopic.\{NEWS\}, hasLength.\{LONG\}, \dots\}, \\ &\dots\}\end{aligned}$$

A preference is context-independent, and thus applicable to any context, if $\mathcal{C}$ consists of all possible situations.

In the previous example, the preferred topic of documents is a constant (*NEWS*), but sometimes a preference varies with the context. For instance,

“*prefer documents that are written by colleagues in my room*”. In this case, the preferred documents depend on whom the user is with at that moment. In this case, we use a variable v to denote it. The rule is $(\mathcal{C}, \mathcal{I})$ with:

$$\begin{aligned}\mathcal{C} &= \{PETER\} \sqcap (\exists hasColleague.((\exists hasRoom.(\exists roomOf.\{PETER\})) \sqcap v)) \\ \mathcal{I} &= Document \sqcap (\exists hasAuthor.v)\end{aligned}$$

We call this kind of preferences **variable based context-aware query preference**, and the former **constant context-aware query preference**.

Other considerations

One of the other things to consider is that, next to the subsumption hierarchy, an ontology often consists of so-called part-whole relations/hierarchies, e.g., a minute is part of an hour, a room is part of a building, and a person can be part of a family. Sometimes, inference over these hierarchies is necessary, for example, if we want to state that if a door is part of a room and a room is part of a house, this door is part of the house. Since research on part-whole relations is a whole field in itself (mereology), we will not address these issues within this thesis. However, one is pointed to the work of Artale et al. (1996), Rector (2003), and Borgida and Brachman (2003), for discussions on mereology and its relation to Description Logics. Practical methods to deal with these hierarchies are given by the “Semantic Web Best Practices and Deployment Working Group” (OWL Part-whole, 2008).

Finally, since OWL gets support for rule languages, for example by the OWL rule language (ORL) as described by Horrocks and Patel-Schneider (2004), the question arises how our preference rules relate to their forms. Although ultimately these languages might incorporate the features that are necessary for dealing with context-aware preference rules, currently (around 2007) they still lack the support for uncertainty and possibilities to attach strengths to the rules.

3.5 Addressing uncertainty in CIM

In the previous section we concluded that whether a document satisfies a preference rule in a situation is based on instance membership. When the sensing of context or documents introduces uncertainty then membership of an instance a in a context $\mathcal{C}$ is no longer a truth-value (true, false) but rather a probabilistic value in $[0 \dots 1]$; denoted as $P(\mathcal{C}(a)) \in [0, 1]$.

In this section, we will therefore discuss how to address data uncertainty when using our model. As the representation language of our model is De-

scription Logics, we first review existing approaches on addressing uncertainty in Description Logics in Section 3.5.1. Concluding that most existing work in the area of Description Logics focuses on domain uncertainty instead of data uncertainty, we present in Section 3.5.2 our approach based on the work of Fuhr and Rölleke (1997) to address membership as a probabilistic event. To separate the user representation from the data uncertainty, we introduce the distinction between the *specifications* of documents and situations (the model) and their *real* counterparts. The uncertainty is in the mapping between the two.

The effect of this approach to compute the membership probabilities is given in Section 3.5.3. To keep the probability computations feasible, we assume that, in general, base events are independent. However, in some cases the Description Logics ontology may explicitly state dependency between base events. In these cases we address *disjointness*, *identicalness*, and *inclusion* of events as discussed in Section 3.5.4.

Finally, we show in Section 3.5.5 how to extend the expressiveness of our model by introducing vague roles and concepts; concepts and roles for which the membership is probabilistic by definition.

3.5.1 Uncertainty in Description Logics

Previous work on probabilistic reasoning in Description Logics (Koller et al., 1997; Jaeger, 1994; Giugno and Lukasiewicz, 2002) mainly focused on defining probabilities on domains. An example of such kind of probability is: “*The probability that a random chosen student is drinking coffee is greater than 0.3*”, or as a probabilistic assertion:

$$P(\text{DrinkingCoffee}|\text{Student}) > 0.3$$

These assertions state statistical information and usually result from an experiment or trial. In our research, however, we are interested in probabilities on assertions coming from sensors, i.e., the so-called *degree of belief*. An example of such a probability is: “*The probability that Eric (a particular student) is drinking coffee is 0.7*”; or as a probabilistic assertion:

$$P(\text{DrinkingCoffee}(\text{ERIC})) = 0.7$$

To explain the meaning of such assertions, we stipulate that there are multiple possible worlds and a probability over these possibilities; properties may hold in some worlds and not in others. The assertion above then says that the total probability of the set of possible worlds in which Eric is drinking coffee is 0.7. Baader et al. (2003b) gives a good overview of various approaches to deal with uncertainty using Description Logics.

3.5.2 Representing uncertainty with events

The uncertainty that we want to model is not concerned with the domain (e.g., there is no uncertainty about whether an eating activity is a free time activity), but only with the uncertainty of membership (e.g., there is uncertainty about whether Eric is eating). Moreover, we adhere to the *closed world assumption*; the instance data is assumed to be complete (see also Section 3.6.1).

Since membership checking over complete data bears similarity to database querying (see the work of Borgida et al. (2003), and the discussion in the next section), the challenges for addressing uncertainty are similar to the field of probabilistic databases. For this reason, we explore the probabilistic database techniques to express the uncertainty of concept and role memberships.

In the database field, there exist two ways for dealing with uncertainty in the data models, namely, *extensional semantics* and *intensional semantics*. The extensional semantics approach is to modify the operators of the algebra to compute probabilities. The problem of this approach is that it ignores most correlations between intermediate results and may give different results depending on the query plan (Dalvi and Suciu, 2004a).

The above problems are circumvented by the intensional semantics approach, which keeps track of the events that contribute to a derived fact, using a so-called *event expressions*, and determine the probability of this derived result (Fuhr and Rölleke, 1997). The intensional semantics approach is sometimes considered impractical, since the number of event expressions can become very large and calculating probabilities of the event expressions takes exponential time. However, some effective optimization techniques have already been developed such as exploiting independency between parts of the computation as suggested by Sarma et al. (2007) and approximation using Monte Carlo algorithms as suggested by Dalvi and Suciu (2004a). Furthermore, keeping track of the events allows us to do explanation of probabilistic query answers, as we discuss in Chapter 5. For these considerations, we decide to follow the same line as the intensional semantics.

The basic idea of intensional semantics is to view each piece of instance data in the knowledge base as a probabilistic event. A (probabilistic) event is a set of possible worlds. For example, the probabilistic assertion from the previous section *DrinkingCoffee(ERIC)* can either be seen as an event, or as the set of all possible worlds in which Eric is drinking coffee. The negation of an event is its complement with respect to all possible worlds.

To address uncertainty in CIM, we postulate the sets Doc_{real} and Sit_{real} consisting of abstractions of real documents and situations. Furthermore we postulate Doc_{spec} and Sit_{spec} consisting of abstractions of *specifications* of

documents and situations (e.g., if there are n properties of documents, there are 2^n document specifications). Each real document and situation is connected to one or more specifications, with a probability, through a function $P(d, d') \in [0..1]$. The intuition is that this probability is a measure of the certainty that d conforms to the specification of d' . We assume that, although context data is uncertain, the uncertainty is complete and consistent, meaning that if we consider all specifications corresponding to one real document or situation, the probability is one:

$$\begin{aligned} \forall d \in Doc_{real} \quad 1 &= \sum_{d' \in Doc_{spec}} P(d, d') \\ \forall s \in Sit_{real} \quad 1 &= \sum_{s' \in Sit_{spec}} P(s, s') \end{aligned}$$

As an example, for a “rather red and thick” document d , its relation to the possible specifications might be as follows:

d'	$P(d, d')$
red, thick	0.9
not red, thick	0.04
red, not thick	0.05
not red, not thick	0.01

and therefore $\sum_{d' \text{ that stands for red}} P(d, d') = 0.95$. Especially descriptions, in form of contexts, interests, or queries are expressed as properties of specifications. We therefore define the probability for the membership of an instance d in a concept Q as:

$$P(d \in Q) = \sum_{d' | Q(d')} P(d, d') \quad (3.2)$$

Preference rules are defined in terms of specifications. The uncertainty comes about when matching specifications to real counterparts. Under these constraints, the preference of a real document d_1 over a real document d_0 in a real situation s ($d_0 \prec_s d_1$) follows from the preference rule for the related specifications of documents and situations:

$$\begin{aligned} P(d_0 \prec_s d_1) &= \\ &\sum_{d'_0, d'_1 \in Doc_{spec}, s' \in Sit_{spec} | d'_0 \prec_{s'} d'_1} P(s, s') \cdot P(d_0, d'_0) \cdot P(d_1, d'_1) \end{aligned}$$

3.5.3 Computing instance membership of concepts

Recall that both the applicability of the preference rule, and whether a document satisfies an interest can be decided by determining the members of a Description Logics concept. By introducing uncertainty, membership becomes uncertain as we saw in Equation (3.2). As discussed in the previous section, we will associate an event with each membership.

Table 3.1 – Description Logics role and concept memberships with basic event identifiers e and basic probabilities β .

(a) Concept membership

Concept membership	Basic event e	$\beta(e)$
$Person(ERIC)$	$\top_e$	1
$Person(PETER)$	$\top_e$	1
$Person(MAARTEN)$	$\top_e$	1
$EatActivity(LUNCH)$	$\top_e$	1
$EatActivity(COFFEEPLUS)$	$\top_e$	1
$Room(ROOM3061)$	$\top_e$	1
$Room(ROOM4061)$	$\top_e$	1
$Room(COFFEEROOM)$	$\top_e$	1

(b) Role membership

Role membership	Basic event e	$\beta(e)$
$hasActType(ERIC, LUNCH)$	$hAc1$	0.3
$hasActType(MAARTEN, LUNCH)$	$hAc2$	0.7
$hasActType(ERIC, COFFEEPLUS)$	$hAc3$	0.8
$hasRoom(ERIC, ROOM3061)$	$hRo1$	0.4
$hasRoom(ERIC, ROOM4061)$	$hRo2$	0.3
$hasRoom(ERIC, COFFEEROOM)$	$hRo3$	0.3

For expressing these events, there is a distinction between a *basic event* and a *complex event*. The event expressions of membership of an atomic concept $e(D(a))$ or role $e(R(a, b))$, as shown in Table 3.1, are basic events. These basic events are identified by an event key. The membership of (derived) Description Logics concept or role expressions, is identified with a *complex event*, denoting a Boolean combination of basic events. The Description Logics special top concept $\top$ and bottom concept $\perp$ correspond to the special basic events $\top_e$ (denoting a certain event) and $\perp_e$ (denoting an impossible event), respectively.

We introduce a recursive function e which maps a Description Logics

Table 3.2 – Recursive definition of function e , mapping a Description Logics membership expression x to an event expression $e(x)$.

DL membership x	event expression $e(x)$
$D(a)$	basic event - see Table 3.1a
$R(a, b)$	basic event - see Table 3.1b
$\top$	$\top_e$
$\perp$	$\perp_e$
$\{b_1, \dots, b_n\}(a)$	$\top_e$, if $a \in \{b_1, \dots, b_n\}$ $\perp_e$, otherwise
$(\neg D)(a)$	$\neg e(D(a))$
$(D \sqcap E)(a)$	$e(D(a)) \wedge e(E(a))$
$(D \sqcup E)(a)$	$e(D(a)) \vee e(E(a))$
$(\exists R.D)(a)$	$\bigvee_b (e(R(a, b)) \wedge e(D(b)))$
$(\forall R.D)(a)$	$\neg(\bigvee_b (e(R(a, b)) \wedge \neg e(D(b))))$
$\exists_a D(a)$	$\bigvee_a e(D(a))$

concept expression to an event expression. The definition of the function is described in Table 3.2. Using this function, we can obtain the event expression for the instance membership for a Description Logics concept expression.

Similar to the probabilistic relation algebra from Fuhr and Rölleke (1997) only the probabilities of basic events are given explicitly via a *basic event probability assignment function* β , satisfying the conditions

1. $\beta(\perp_e) = 0$.
2. $\beta(\top_e) = 1$.
3. $0 \leq \beta(e) \leq 1$, for all basic events e

From these, probabilities of complex events are computed. In general, probabilities for event expressions are given by a general probability assignment function P which maps an event expression to a value in $[0, 1]$.

Example 3.5.1. Suppose we have a preference rule which is applicable if “Eric is eating either in his own room (room 3061) or in the coffeeroom”:

$$\begin{aligned} \mathcal{C} = \{ERIC\} \sqcap \exists hasActType.EatActivity \\ \sqcap \exists hasRoom.\{COFFEEROOM, ROOM3061\} \end{aligned}$$

To assess whether the preference rule is applicable in the situation given by Table 3.1, we have to determine whether there is at least one instance a that is a member of $\mathcal{C}$. Since it is trivial to see that only the instance $ERIC$ can match this context, we will show how to determine $e(\mathcal{C}(ERIC))$. The definition of e function from Table 3.2 implies:

$$\begin{aligned}
e(\mathcal{C}(ERIC)) = & \\
& \left(\begin{cases} \top_e & \text{if } ERIC \in \{ERIC\} \\ \perp_e & \text{otherwise} \end{cases} \right) \\
& \wedge \bigvee_b (e(\text{hasActType}(ERIC, b)) \wedge e(\text{EatActivity}(b))) \\
& \wedge \bigvee_c (e(\text{hasRoom}(ERIC, c)) \\
& \quad \wedge \left(\begin{cases} \top_e & \text{if } c \in \{COFFEEROOM, ROOM3061\} \\ \perp_e & \text{otherwise} \end{cases} \right))
\end{aligned}$$

For the data from Table 3.1 this results in the following event:

$$\begin{aligned}
e(\mathcal{C}(ERIC)) = & \top_e \\
& \wedge ((hAc1 \wedge \top_e) \vee (hAc3 \wedge \top_e)) \\
& \wedge ((hRo1 \wedge \top_e) \vee (hRo2 \wedge \perp_e) \vee (hRo3 \wedge \top_e)) \\
= & (hAc1 \vee hAc3) \wedge (hRo1 \vee hRo3)
\end{aligned}$$

For the sake of concreteness, assume temporarily that tuples are, contrary to what is stated in the ontology, independent of each other, that is, $P(e_1 \vee e_2) = P(e_1) + P(e_2) - P(e_1) \cdot P(e_2)$ and $P(e_1 \wedge e_2) = P(e_1) \cdot P(e_2)$. Then the probability of the above event expression can then be determined directly from the basic probabilities in Table 3.1:

$$\begin{aligned}
P(e(\mathcal{C}(ERIC))) &= P((hAc1 \vee hAc3) \wedge (hRo1 \vee hRo3)) \\
&= (0.3 + 0.8 - 0.3 \cdot 0.8) \cdot (0.4 + 0.3 - (0.4 \cdot 0.3)) \\
&= 0.4988
\end{aligned}$$

□

3.5.4 Ontological dependencies in probability computation

In the example of the previous section, we ignored dependencies during the computation of the probability of instance membership. However, since we consider the domain knowledge to be certain, the probabilities of Description Logics instance concept and role memberships should satisfy the following properties:

Domain knowledge	Consequence, for arbitrary a, b
$C \sqsubseteq D$	$P(e(C(a))) \leq P(e(D(a)))$
$C \equiv D$	$P(e(C(a))) = P(e(D(a)))$
$R \equiv S$	$P(e(R(a, b))) = P(e(S(a, b)))$
$R \sqsubseteq S$	$P(e(R(a, b))) \leq P(e(S(a, b)))$

where C and D are Description Logics concepts, R and S are Description Logics roles, and a and b are Description Logics concept instances.

For example, consider the exact number restriction from the domain knowledge in Figure 3.5:

$$Person \sqsubseteq \exists_1 hasRoom.Room$$

Then we have $P(e((\exists_1 hasRoom.Room)(ERIC))) = 1$ since:

$$\begin{aligned}
& 1 \\
& = \quad \text{Table 3.1a} \\
& \quad P(e(Person(ERIC))) \\
& \leq \quad \text{Domain knowledge, } Person \sqsubseteq \exists_1 hasRoom.Room \\
& \quad P(e((\exists_1 hasRoom.Room)(ERIC))) \\
& \leq \quad \text{Property of probability} \\
& 1
\end{aligned}$$

This means that, independent of the amount of uncertainty that the data possesses, the membership probability of instance *ERIC* in the concept $\exists_1 hasRoom.Room$ is exactly one. When multiple sensors give conflicting context data, this information can facilitate estimation of probabilities.

Given the fact that many important properties in the domain of context-awareness are usually dependent of each other (e.g., a person can only be at a single location at one time), it is necessary to take into account the correlations among basic events. Table 3.3 illustrates the influence of correlations between two events on probability calculation. It is adapted from Suciu and Dalvi (2005).

Often these correlations are determined from statistical analysis, which is impractical to do when deploying a context-aware system. However, many useful dependencies for context-awareness can already be acquired from the ontology using inference. In our study, we consider the following four event correlations:

Disjointness: When two events by definition cannot happen at the same time. E.g., if someone is in a certain room, she cannot be in another room at the same time.

Table 3.3 – Event correlation vs. probability between events e and e' , adapted from Suciu and Dalvi (2005).

Correlation	Probability
Disjointness	$P(e \wedge e') = 0$
Negatively correlated	$P(e \wedge e') < P(e) \cdot P(e')$
Independent	$P(e \wedge e') = P(e) \cdot P(e')$
Positively correlated	$P(e \wedge e') > P(e) \cdot P(e')$
Identical	$P(e \wedge e') = P(e) = P(e')$

Inclusion: A special kind of positive correlation where in case one event happens, by definition, the other happens as well. E.g., if someone is in a room, then she is also located in the building which contains this room.

Identicalness: When two events by definition always happen together. E.g., one could define that a person is sailing, if and only if she is in a sailing boat. In this situation, the event *sailing* and *being in a sailing boat* are identical.

Independence: When the occurrence of one event does not say anything about the occurrence of the other. E.g., the weather is (probably) not dependent on the day of the week.

For practical purposes, we only consider dependencies between two events. Although it is possible that the ontology defines dependencies over more than two events, this would require us to compare a maximum of $\frac{1}{2}2^n$ combinations for each event expression of n events, instead of the maximum of $\binom{n}{2}$ when only considering dependencies between two events.

In conclusion, for computing the membership probability we take *disjointness*, *inclusion*, and *identicalness* between two basic events into account, in case that the dependence can be inferred from the domain knowledge. In all other cases we assume *independence* between events.

In practice, the correlation between two basic events is determined using a Description Logics reasoner. We therefore construct for both basic events a concept which *retrieves* exactly one instance with the corresponding event. For the two concepts, we can query the reasoner whether they are disjoint, identical, or included. For example, for the events *hRo1* and *hRo3* from

Figure 3.6 – Graphical representation of two correlated basic events e and e' , and an event expression X .

Table 3.1b, we could ask the reasoner whether the following concepts C_1 and C_2 are disjoint:

$$C_1 \equiv \{ERIC\} \sqcap \exists hasRoom.\{ROOM3061\}$$

$$C_2 \equiv \{ERIC\} \sqcap \exists hasRoom.\{COFFEEROOM\}$$

The first concept retrieves the instance *ERIC* with the basic event *hRo1*, the second concept retrieves also the instance *ERIC*, but now with the event *hRo3*. The concepts are disjoint indeed, since according to the ontology a person can only be at one room ($Person \sqsubseteq \exists_1 hasRoom.Room$) and we adhere to the *unique naming assumption*, which means that *ROOM3061* is different from *COFFEEROOM*.

Although calculating these correlations seems impractical for a large number of instances and events, we expect optimization strategies such as the pre-calculation of disjoint events or addressing multiple dependent events at the same time to speed up this process. In this chapter, we only show that proper dependency calculation is possible.

Given that the Description Logics reasoner can find the correlation between two events, we have to address the influence of their correlations on the calculation of event probabilities. For *conjunctions* of literals with basic events, these correlations can be taken into account by comparing each event with all other events in the conjunction: let e and e' be two basic events, and let X be either a basic or complex event expression, the implications of correlation for the probability of the conjunctions, by straightforward application of basic probability theory, are then as follows :

First, in case event e is disjoint from event e' (see Figure 3.6a), the following equations hold:

$$P(X \wedge \neg e \wedge \neg e') = P(X) - (P(X \wedge e') + P(X \wedge e)) \quad (3.3)$$

$$P(X \wedge \neg e \wedge e') = P(X \wedge e') \quad (3.4)$$

$$P(X \wedge e \wedge \neg e') = P(X \wedge e) \quad (3.5)$$

$$P(X \wedge e \wedge e') = 0 \quad (3.6)$$

If event e includes event e' (see Figure 3.6b), the following equations hold:

$$P(X \wedge \neg e \wedge \neg e') = P(X \wedge \neg e)$$

$$P(X \wedge \neg e \wedge e') = 0$$

$$P(X \wedge e \wedge \neg e') = P(X) - P(X \wedge e) + P(X \wedge e')$$

$$P(X \wedge e \wedge e') = P(X \wedge e)$$

Finally, if the events e and e' are identical, the following equations hold:

$$P(X \wedge \neg e \wedge \neg e') = P(X \wedge \neg e)$$

$$P(X \wedge \neg e \wedge e') = 0$$

$$P(X \wedge e \wedge \neg e') = 0$$

$$P(X \wedge e \wedge e') = P(X \wedge e)$$

Since at the right hand side of each equation, the correlated events no longer exist in the same conjunctions, these equations are used to remove all ontology related dependencies that we are concerned with. As a result, the remaining conjunctions only contain basic events that we consider independent. For these conjunctions, the probability is equal to the multiplication of the probabilities of their basic events.

If an event expression also contains disjunctions, we convert it to its equivalent disjunctive normal form, with only complete conjunctions. The *disjunctive normal form* (DNF) for an event is a disjunction consisting of one or more conjunctions of one or more literals where the not operator can only be used as part of a literal (e.g., $(e_1 \wedge e_2 \wedge e_3) \vee (e_1 \wedge \neg e_2 \wedge e_3) \vee (e_1 \wedge e_2 \wedge \neg e_3)$). A *complete conjunction* is a conjunction which contains each basic event exactly once, either in positive or in negative form (Fuhr and Rölleke, 1997).

In a DNF with complete conjunctions, all conjunctions are disjoint and the probability of the complex event is equal to the sum of probabilities of the different conjunctions. For these conjunctions, we can address the ontological dependencies using the equations given before.

Let us illustrate how to apply this method with an example:

Example 3.5.2. Suppose we again want to compute the probability of the event that “Eric is eating either in his own room (room 3061) or in the coffeeroom,” but now exploiting the dependencies that result from the ontology in Figure 3.5. Recall that the event was $(hAc1 \vee hAc3) \wedge (hRo1 \vee hRo3)$.

The only dependency from the ontology that is relevant for computing the probability of this expression is the disjointness of $hRo1$ and $hRo3$, resulting from the statement $Person \sqsubseteq \exists_1 hasRoom.Room$. As indicated in the beginning of this section, we consider other events to be independent.

Rewriting the event expression to its DNF with complete conjunctions and adding up the probabilities of the conjunctions gives:

$$\begin{aligned}
 & P((hAc1 \vee hAc3) \wedge (hRo1 \vee hRo3)) \\
 &= P(hAc1 \wedge hRo3 \wedge hAc3 \wedge hRo1) \\
 &\quad + P(hAc1 \wedge hRo3 \wedge hAc3 \wedge \neg hRo1) \\
 &\quad + P(hAc1 \wedge hRo3 \wedge \neg hAc3 \wedge hRo1) \\
 &\quad + P(hAc1 \wedge hRo3 \wedge \neg hAc3 \wedge \neg hRo1) \\
 &\quad + P(hAc1 \wedge hRo1 \wedge hAc3 \wedge \neg hRo3) \\
 &\quad + P(hAc1 \wedge hRo1 \wedge \neg hAc3 \wedge \neg hRo3) \\
 &\quad + P(hAc3 \wedge hRo1 \wedge \neg hAc1 \wedge hRo3) \\
 &\quad + P(hAc3 \wedge hRo1 \wedge \neg hAc1 \wedge \neg hRo3) \\
 &\quad + P(hAc3 \wedge hRo3 \wedge \neg hAc1 \wedge \neg hRo1)
 \end{aligned}$$

Conjunctions where the disjoint terms $hRo1$ and $hRo3$ occur in positive form evaluate to zero according to Equation 3.6. Furthermore, we can apply Equation 3.4 and 3.5 to remove the conjunctions in which one of the disjoint terms ($hRo1$ and $hRo3$) appears in positive form and the other one in negative form. This leads to the following equation:

$$\begin{aligned}
 & P((hAc1 \vee hAc3) \wedge (hRo1 \vee hRo3)) \\
 &= P(hAc1 \wedge hRo3 \wedge hAc3) \\
 &\quad + P(hAc1 \wedge hRo3 \wedge \neg hAc3) \\
 &\quad + P(hAc3 \wedge hRo3 \wedge \neg hAc1) \\
 &\quad + P(hAc1 \wedge hRo1 \wedge hAc3) \\
 &\quad + P(hAc1 \wedge hRo1 \wedge \neg hAc3) \\
 &\quad + P(hAc3 \wedge hRo1 \wedge \neg hAc1)
 \end{aligned}$$

In the resulting equation, there are no more ontology-related correlations between two events. Because we assume independence among the rest of the

events, we can multiply the probability of the basic events contained in the conjunctions resulting in a total probability of 0.602. This result is, as expected, more than the result given in the previous section, where independence was assumed among all the basic events. And, since this result takes into account the explicitly specified dependencies, it is preferred to the previous result. $\square$

Note that although the probability of membership does lead to a sort of weight for the preference of a document, it is different from a possible weighting or scoring of the preference rule itself. Consider, for example, the preference for news documents in the previous section. It could indeed be the case that some text classification system concludes that a document is about news with a probability of 0.8. In this case, a document that is certainly about news is preferable, but this is unrelated to the strength of the user's preference for news articles.

3.5.5 Vague roles and concepts

Description Logics are quite expressive; they enable to describe concepts in a way that facilitates reasoning and are supported by various modeling tools. The same logic description, however, has as downside that some complicated preferences might not be expressible using Boolean operators. Especially when considering non-discrete properties such as 'around', time concepts such as 'soon', or location concepts such as 'nearby'.

Often, these cases can be solved using discretization, where 'soon' could for example be defined as "within five minutes". However, in some situations this is not an adequate solution. For these cases, we propose a solution based on the work of Fuhr and Rölleke (1997), namely, the concept of vague roles and concepts.

Definition 3.5.1. *A concept or role is vague if the membership is probabilistic and the probabilities for membership are determined by an external function.*

This external function could for example have a Gaussian or logarithmic behavior, depending on the attributes of the included or related concepts.

As an example, suppose the user wants to present a preference for cheap holidays. The user can construct this interest in the following three ways:

$$\begin{aligned}\mathcal{I} &= \textit{CheapHolidays}, \\ \mathcal{I} &= \textit{Holiday} \sqcap \textit{Cheap}, \text{ and} \\ \mathcal{I} &= \textit{Holiday} \sqcap \textit{hasPriceType}.\{\textit{CHEAP}\}.\end{aligned}$$

In the first case, instances are members of the vague concept *CheapHolidays*, with a probability stating to which extent the user considers the instance to be a cheap holiday.

In the second case, the concept *Cheap* is vague, and includes as members all instances with prices that the user considers to be cheap. The membership probability of a certain instance for this concept could, for example, be a logarithmic function based on the price.

In the third construction, the role *hasPriceType* is vague, and might be dependent on the product and the price. For example, the membership *hasPriceType*(*HOLIDAYSPAIN123*, *CHEAP*) might have a basic probability of 0.99, if the price of *HOLIDAYSPAIN123* is 200 Euro. Whereas the membership *hasPriceType*(*COFFEE123*, *CHEAP*) might have a (basic) probability of 0.01, if the price of *COFFEE123* is “only” 10 Euro.

Note that for such construction as ‘around’ with concrete numbers, one would have to create a specific vague role or concept for the value that the price should be around, e.g., *AroundTen*.

Finally, one might argue, that we could model all the preferences of a user, using only one vague concept, as follows:

$$\mathcal{I} = \textit{InterestingThings}$$

Although there are no formal objections to such a model, it would clearly violate the demands of traceability and reasoning for our modeling language.

3.6 Implementation and usage of CIM

In this section we show how the extra knowledge of the user’s information need is exploited in query processing. Therefore we first present the realization of our model in the domain of relational databases in Section 3.6.1. After that, we discuss two access modes, namely, *pull query* where users actively query databases to pull relevant information, and *push query* where the systems push proactively possibly relevant information to users (e.g., querying the background information about a person when she enters a room) in Section 3.6.2.

In the present approach, we are not so much concerned with efficiency but with applicability and feasibility. The approach of this chapter has been implemented as a prototype in PostgreSQL (see also Section 4.6).

Person	Room	Paper	EatActivity	hasActivityType		hasAuthor	
ID	ID	ID	ID	SOURCE	DESTINATION	SOURCE	DESTINATION
ERIC	ROOM3061	DOC1	LUNCH	ERIC	LUNCH	DOC1	ERIC
PETER	ROOM4061	DOC2	COFFEEPLUS	PETER	LUNCH	DOC1	PETER
MAARTEN	COFFEEROOM	DOC3	DINER	MAARTEN	COFFEEPLUS	DOC2	MAARTEN
...	...	...	...	...	...	...	...

hasFriend		hasRoom		hasTopic	
SOURCE	DESTINATION	SOURCE	DESTINATION	SOURCE	DESTINATION
ERIC	PETER	ERIC	ROOM3061	DOC1	NEWS
PETER	ERIC	PETER	ROOM3061	DOC2	XML
PETER	MAARTEN	MAARTEN	COFFEEROOM	DOC1	SPORT
...	...	...	...	...	...

Figure 3.7 – Role and concept tables.

3.6.1 Implementation of CIM

To integrate context-aware preferences with database queries, we need to provide a way in which can relate both context and interest concepts to database relations. For the interest concept, this is a rational translation, since the interest affects the query. For the context concept it is less evident, since dynamic contexts (e.g., location, surrounding people, etc.) must be acquired real-time from external sources/services like sensor networks. However, it is in line with the efforts of the sensornet community which has embraced declarative queries as a key programming paradigm for large sets of sensors (Deshpande et al., 2004). Furthermore, storage and logging context implies having an underlying storage system that provides this view. Here, we take the SQL query language as a uniform interface to the context data.

The relation of Description Logics to databases is given in a good survey of Borgida et al. (2003). Examples of previous mappings of Description Logics to databases include the work of Horrocks et al. (2004), who describe a Description Logics instance store to provide an infrastructure for reasoning with a large number of instances. Furthermore, Borgida and Brachman (1993) describe a method for loading data in Description Logics reasoners, where concepts and role views are introduced to process Description Logics queries over the database. Transformations of Description Logics to other types of databases (deductive, object-oriented) have been done as well (Weithoner et al., 2003; Peim et al., 2002).

Similarly to Borgida and Brachman (1993), we represent each concept by a table which has the concept name as the table name, and each table has one attribute: *ID*. The tuples of the table represent the instances of the concept. An extra table *TOPTABLE* represents the instances in the domain. Also, we represent each role by a table, with the role name as its table name and containing two attributes *SOURCE* and *DESTINATION*. For each tuple of the table, the role relates the *SOURCE* instance with the *DESTINATION*

Table 3.4 – Mapping Description Logics concept expressions $expr$ to SQL query $\llbracket expr \rrbracket$, where D, E are concept expressions, R is a role, C is an atomic concept, and a is an instance.

DL $expr$	SQL $\llbracket expr \rrbracket$
C	SELECT ID FROM C
a	VALUES ('a')
$\top$	SELECT ID FROM TOPTABLE
$\perp$	NULL
$\neg D$	(SELECT ID FROM TOPTABLE) EXCEPT (SELECT ID FROM $\llbracket D \rrbracket$)
$D \sqcap E$	(SELECT ID FROM $\llbracket D \rrbracket$) INTERSECT (SELECT ID FROM $\llbracket E \rrbracket$)
$D \sqcup E$	(SELECT ID FROM $\llbracket D \rrbracket$) UNION (SELECT ID FROM $\llbracket E \rrbracket$)
$\exists R.D$	SELECT R.SOURCE FROM R WHERE R.DESTINATION IN (SELECT ID FROM $\llbracket D \rrbracket$)
$\forall R.D$	(SELECT ID FROM TOPTABLE) EXCEPT (SELECT R.SOURCE FROM R WHERE R.DESTINATION IN ((SELECT ID FROM TOPTABLE) EXCEPT (SELECT ID FROM $\llbracket D \rrbracket$)))

instance. Figure 3.7 gives examples of this method. If we want to query existing databases that do not have this schema, one can construct concept and role views on top of the existing structure.

Table 3.4 shows how we express Description Logics concept expressions using SQL queries, as adapted from Borgida and Brachman (1993).

Support for uncertainty

Incorporating uncertainty into the database realization is rather trivial. In fact, if we apply the mapping to SQL from Table 3.4, we can use the existing approach from Fuhr and Rölleke (1997) to arrive at the resulting event expression. This resulting event expression is similar to the one from Section 3.5.2, and we can use the same techniques as in Section 3.5.4 for addressing ontological dependencies. In our prototype, we introduced functions for the Description Logics constructors that combined the selection of instances from Table 3.4 with the events from Table 3.2 in a resulting view.

Reasoning and databases

To keep the reasoning part simple we assume that the database contains the complete (probabilistic) information about all instances, and that the data in the database adheres to the ontology.

The first assumption allows us to do so-called *negation by failure* in Table 3.4; if an assertion is not in the database, it is not true (closed world assumption). This is in contrast with the open world assumption (OWA)

normally used in Description Logics, that states that absence indicates lack of knowledge instead of being untrue. The second assumption allows us to use the ontology for dependency calculations.

How the database internally handles incomplete data or data that does not adhere the ontology is out of the scope of this thesis. A possible solution to keep the data consistent with the ontology is to use triggers. For example, if the ontology defines that if someone is eating, she is also doing a freetime activity, whenever an instance is inserted into the table of the *EatActivity* concept in Figure 3.5, one could fire a trigger that would insert the same instance into the *FreeTimeAct* relation, the *ActType* relation, and the *Thing* relation. However, the implications of such method are not trivial.

3.6.2 Usage of CIM

Context-aware preferences can assist two kinds of database accesses: pull and push. First, in the *pull* access mode, context-aware preferences are used for query augmentation (e.g., enforcing the query constraint *topic*=“XML” to the user’s query over documents when she is doing some free time activity with some friends in the same room). Second, in the *push* access mode, context-aware preferences are used as query triggers (e.g., retrieving the background information about a person when she is nearby).

Let a query $q_{user} \subseteq Doc$ denote the set of documents that are described by the user’s query, and $db \subseteq Doc$ the documents that are stored in the database. Figure 3.8 shows the pull-push query execution framework equipped with CIM. It contains six major components. 1) The *context supplier* is responsible for supplying the current query context for the current situation, $s \in Sit$. Some static contexts like user’s background, friends, documents, etc. are obtained from *context database*; while some dynamic contexts like user’s location, emotion, traffic, surrounded people, etc. are obtained from sensors or external service providers. 2) The *preference selector* selects from the *preference repository* relevant context-aware query preferences, if necessary by reasoning. A context-aware preference $(\mathcal{C}, \mathcal{I})$ is relevant if it is applicable in the current situation and the interest can be mapped to certain relations included in the user’s query, $\mathcal{I} \cap q_{user} \neq \emptyset$, (pull query) or included in the database, $\mathcal{I} \cap db \neq \emptyset$, (push query). 3) In the pull mode, the *query adapter* augments user’s query q_{user} with the relevant preference, and optimizes it further into q'_{user} . 4) In the push mode, the *query trigger* proactively generates a query $q_{trigger}$ according to the preference, and sends it to the underlying DBMS for execution. 5) The *access log* records user-system interactions, from which the *preference miner* can discover the user’s context-aware preferences. Users can also directly input their preferences. 6) The *preference*

Figure 3.8 – The preference aware pull-push query framework.

manager is responsible for storing, maintaining, and managing the user's context-aware preferences.

Given a rule $(\mathcal{C}, \mathcal{I})$ expressed in Description Logics, the SQL query $\llbracket C \rrbracket$ (see Table 3.4) is applied by the system in Figure 3.8 to the context database; a non-empty query result implies that the current context is consistent with $\mathcal{C}$. The associated interest $\mathcal{I}$ is then activated for either query adaptation or query trigger (where $\mathcal{I}$ is translated into SQL as a proactive push query).

Pull queries

For pull queries, we focus on retrieving only those results that are preferred by the user. As an example, a user raises a pull query for documents:

```
SELECT ID FROM Document
```

The context-aware preference rule $(\mathcal{C}, \mathcal{I})$, with

$$\begin{aligned}\mathcal{C} &= \{PETER\} \sqcap (\exists hasFriend.(\exists hasRoom.(\exists roomOf.\{PETER\}))) \\ \mathcal{I} &= Document \sqcap (\exists hasTopic.\{XML\}),\end{aligned}$$

contains a concept *Document* which appears in the query. To determine whether the preference rule is applicable, the concept $\mathcal{C}$ is translated straightforwardly into $\llbracket C \rrbracket$, which is:

```
SELECT * FROM ((VALUES ('PETER'))
INTERSECT
(SELECT hasFriend.SOURCE FROM hasFriend
WHERE hasFriend.DESTINATION IN
(SELECT hasRoom.SOURCE FROM hasRoom
WHERE hasRoom.DESTINATION IN
(SELECT roomOf.SOURCE FROM roomOf
WHERE roomOf.DESTINATION IN (VALUES ('PETER'))))
)))
```

The query could then be simplified into:

```
SELECT hasFriend.SOURCE
FROM hasFriend, hasRoom, roomOf
WHERE hasFriend.SOURCE = 'PETER' AND
      hasFriend.DESTINATION = hasRoom.SOURCE AND
      hasRoom.DESTINATION = roomOf.SOURCE AND
      roomOf.DESTINATION = 'PETER'
```

When this query returns a non-empty result, the original pull query is augmented with the additional constraint, $\mathcal{I}(d)$, in the WHERE clause:

```

SELECT ID FROM Document
WHERE ID IN
  (SELECT hasTopic.SOURCE FROM hasTopic WHERE
    hasTopic.DESTINATION IN (VALUES ('XML'))))

```

In case the underlying context data is uncertain, the context query $\llbracket \mathcal{C} \rrbracket$ also results in a probability that the preference is rule applicable:

$$P(e(\exists_a \mathcal{C}(a)))$$

Furthermore, if the document attributes are uncertain, the results of the original pull query are probabilistic as well. In the case of independency between context and interest memberships, for a rule r the following score is returned as an indication to which extent a document d is preferred:

$$ResultScore(r, d) = P(e(\exists_a (r.\mathcal{C}(a)))) \cdot P(e(r.\mathcal{I}(d)))$$

Push queries

A context-aware preference may also trigger a push query proactively. For example, consider a preference “*retrieving the publications of the person when she enters room 3061*”:

$$\begin{aligned} \mathcal{C} &= v \sqcap \exists hasRoom.\{ROOM3061\} \\ \mathcal{I} &= \exists hasAuthor.v \end{aligned}$$

From this preference rule, a DB2 query trigger is created as follows:

```

CREATE TRIGGER queryAuthorDocs AFTER INSERT ON hasRoom
REFERENCING NEW AS n FOR EACH ROW
WHEN (n.DESTINATION IN VALUES ('ROOM3061'))
  SELECT SOURCE
  FROM   hasAuthor
  WHERE  DESTINATION = n.SOURCE

```

According to the example database in Figure 3.7, if Peter enters the room, the query result will include *DOC1*.

3.7 Discussion

In this chapter, we presented our model CIM for capturing the user’s information need to realize personalized information delivery and dissemination in an *Ubiquitous Computing* environment.

CIM encompasses various types of properties (Section 3.3.1), consists of preference rules that indicate for each situation, the interests of the user

(Section 3.3.3). To support the implications of using context data, contexts and interests are represented as Description Logics concepts (Section 3.4.2).

We showed how CIM is able to cope with uncertainty of context data using event expressions (Section 3.5), and how existing database technology can be used by implementing the model on top of a DBMS (Section 3.6.1). Finally, we showed how CIM can be used to provide personalized query answers and trigger proactive personalized information delivery (Section 3.6.2).

We did make the simplification that user has only a single preference rule r . And although this preference rule could in principle capture the whole information need of a user using a vague concept (as seen in Section 3.5.5), it is not the intended use of the model. For this reason, we will extend our model in the next section with the ability to combine preferences.

Future research should address possible scalability issues and implications of implementing CIM upon different data models, such as XML.

Combining preferences

In the previous chapter, we have introduced our model, based on preference rules with scores. This chapter shows how to combine preferences by introducing two kinds of probabilistic score semantics. It also shows how to address learning of scores and how the combination of preferences affects the implementation on top of an existing relational database system.

4.1 Introduction

Suppose that a user has a preference for technical documents if she is at a university. In case a system is able to determine whether a document is technical and whether the user is at a university, ranking documents according to this preference is quite straightforward; if the user is at a university, documents that are (more) technical should be ranked higher.

However, users often have multiple preferences. The same user might also have a preference for news documents during lunch. It is not clear from these two preferences whether news documents or technical documents are more preferred during lunch at a university.

Furthermore, suppose that this user, during lunch at a university, poses a query for documents about XML to an underlying information retrieval system. Based on its own model, the underlying system will rank the returned documents according to the extent that they are about XML. It is also not clear from the two preferences how one can rank the query results based on a combination of query relevance and preference satisfaction.

In this chapter we address both issues. First, we review existing methods for the combination of preferences in Section 4.2. Then we relate the influence of context to the way current *information retrieval systems* work (in Section 4.3). The main contribution of this chapter is:

- An operation $\odot$ that *combines* two scores $\sigma(r)$ and $\sigma(r')$ into a score for the composite preference rule $\{r, r'\}$.

Actually, the operation comes in two flavors, namely $\otimes$ and $\oplus$, each of which is based on a particular model of the users behavior.

Operation $\odot$ is the set-variant of the 2-ary operation $\odot$, i.e., $\odot\{a, b, \dots\} = a \odot b \odot \dots$

- A way to acquire scores of preference rules from the history of the interaction of a user with a non-context-aware retrieval system (in Section 4.5).

Throughout we aim at, and let us inspire by, a probabilistic interpretation of what a score for a preference rule means. Based on two possible interpretations we derive the two operations $\otimes$ and $\oplus$.

The chapter ends with some remarks about implementation (in Section 4.6) and further issues (Section 4.7).

4.2 State of art

4.2.1 Preference as utility

Preferences are often discussed in relation to *utility* of choices (Russell and Norvig, 2004). The utility theory is a motivation that preferences of users can in fact be modeled through a score function if they satisfy a set of axioms. This score function is in that case equal to a utility function $U : X \rightarrow \mathbb{R}$, where X is a set of (mutually exclusive) events.

These axioms hold for so-called *rational* agents. However, the question presents itself, whether real users are rational (and therefore their preferences can be modeled by such function). Although the exact answer to this question is still open to debate, it can be shown that users do resolve non-rational inconsistencies if they are pointed out to them (Russell and Norvig, 2004).

Modeling users as rational agents and expressing their preference using a utility function, is not a goal in itself; since the utility function can be arbitrarily complex it does not add much to our understanding of the user. The challenge is to model regularities in the preference of a user.

One of such regularities could be the combination of multiple preferences for single attributes to a preference for a complete document, which is studied in the *multiattribute* utility theory. In general we can represent the utility function as follows:

$$U(x_1 \dots x_n) = f[g_1(x_1), \dots, g_n(x_n)]$$

where $x_1, \dots, x_n$ are attributes of a document. The function f should preferably be a simple multiplicative or additive function. However, in the worst case, f is as complex as the utility function U .

In this chapter we introduce our own functions based on probability theory and independence assumptions. Although these independence assumptions makes the model more comprehensible, it is understood that they do severely limit the expressivity of the score model; we therefore motivate the assumptions in Section 4.4.5.

4.2.2 Database research in combining preferences

In the first work on preference querying by Lacroix and Lavency (1987), preferences are considered either more important than others, or equally important. Difference in importance results in a preference application order, where preferences are filters that only let through tuples that satisfy them. Preferences that return no results, are ignored. The semantics of equally important preferences is that the tuples that satisfy the maximal number of preferences are returned.

Agrawal and Wimmers (2000) introduced a method, to combine preferences by combination operators that are designed by the system designer, allowing much flexibility. Their work is orthogonal to the approach in this chapter, which focuses on which combination operator to choose.

Kießling (2005) introduces three preference combination operators:

Pareto accumulation ($p \otimes p'$) Preferences p and p' are equally important.

Document d dominates d' , only if it dominates d' by one preference and is equal or also dominant for the other preferences:

$$d \prec_{p \otimes p'} d' = (d \prec_p d' \wedge d \preceq_{p'} d') \vee (d \preceq_p d' \wedge d \prec_{p'} d')$$

Prioritized (lexicographic) composition ($p \& p'$) Preference p is more important than p' . Document d dominates d' , if it dominates d' by preference p , or is equal with respect to p , but dominant with respect to p' :

$$d \prec_{p \& p'} d' = d \prec_p d' \vee (d \sim_p d' \wedge d \prec_{p'} d')$$

Numerical preference ($rank_F(p, p')$) Given two preferences p and p' that result in a score, the combining function $F : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ determines the preference for documents; a higher resulting score means a higher preference: $d \prec_{rank_F(p, p')} d' = F(p(d), p'(d)) < F(p(d'), p'(d'))$

Chomicki (2003) considers the same composition operators. Furthermore, both Kießling and Chomicki also consider Boolean composition of preferences.

Lately there has been much research in the area of (efficient implementation of) Skyline queries (Börzsönyi et al., 2001). These queries are comparable to Pareto accumulation over a number of attributes. It has been pointed out by Chomicki (2003) that skyline queries cannot be expressed using score functions. However, according to utility theory, all preferences of rational agents can be expressed by score functions and therefore skyline queries that cannot be expressed by score functions are irrational.

As an example of this irrationality, consider the preference specification from Section 3.2: “*prefer one book tuple to another if and only if their ISBNs are the same and the price of the first is lower.*” the skyline formulation for this query is:

SKYLINE OF ISBN DIFF, PRICE MIN

This preference indicates that, for the following data, book *A* is equally preferred as book *B* and *C*, whereas book *B* is preferred to book *C*.

BOOK(ID,ISBN,PRICE)
(A, 1381-3626, 90)
(B, 1381-3625, 70)
(C, 1381-3625, 80)

As pointed out by Chomicki (2003), this example cannot be expressed by score functions, since a score function is unable to express that book *A* has an equal score as both *B* and *C*, whereas book *B* should have a higher score than book *C*. We show the irrationality of this preference in Figure 4.1, where we have substituted the display of the book record by giving the address where the book can be bought.

Finally, research on mining the history to assist database querying has been looked at before by Agrawal et al. (2003) and Chaudhuri et al. (2004). Especially in Chaudhuri et al. (2004) probabilistic models are used that bear similarity to the conjunctive model that will be introduced in Section 4.4.3, but they do not consider context-awareness or high-level features such as Description Logics concepts.

In this thesis we consider quantitative preferences and preference combinations using score combination functions, similar to the $rank_F$ combination from Kießling (2005). There are three motivations why we chose for scores rather than for a partial ordering. First, the resulting score for a document is an absolute measure of importance. For this reason, the resulting score can be used to decide whether two documents differ greatly in preference, or only a little. In the latter case, a context-aware DBMS may decide to return both documents. Second, using scores throughout the preference

Figure 4.1 – The irrationality of “prefer one book tuple to another if and only if their ISBNs are the same and the price of the first is lower” can be shown by an exchange of information and money between two users; Alice, who has this preference, and Bob trying to exploit the user. In each state Alice is willing to make the proposed trade, whereas in the end Alice will lose money.

computation gives us the semantics for combining the strength of a preference with other factors that influence the relevance of a document, such as importance based on external (query) factors and uncertainty of underlying data. Finally, closely related to the other motivations, it allows us to use a probabilistic interpretation of the scores, through which we can motivate combination functions. The probabilistic interpretation of what a score for a preference rule means is also what differentiates the work in this chapter from previous work on score based preferences in databases.

Example 4.2.1. *As an example of how to use scores to specify the weight of a set of preferences, consider the following preferences relating to an order that Marge wants to place at Homer, as taken from Kießling (2005):*

Preferences of Marge (1) *Marge is interested in notebooks.* (2) *The CPU speed must be at least 2 GHz.* (3) *The order quantity should be around 40, and equally important the main memory capacity should be the highest possible.* (4) *Marge’s favorite manufacturers are Toshiba and Hewlett Packard, which is equally important to her other preferences.*

Preferences of Homer (5) *Homer wants to maximize his profit margin.* *But since he is a fair dealer for Marge, all her customer preferences are more important to him.*

As these preferences are shown as a motivation for the qualitative approach, from Kießling, it is not possible to represent everything using score functions. For example, preferences (1) and (2) should be made explicit in the query.

However, the resulting representation is quite satisfactory, if we identify the following non-context-aware preferences for orders (where it is assumed that an order contains only one kind of notebook):

Preference	$\mathcal{C}$	$\mathcal{I}$	σ
r_3	$\top$	QuantityAround40	0.4
$r_{3'}$	$\top$	$\exists \text{hasMemoryAmount}.\{\text{High}\}$	0.4
r_4	$\top$	$\exists \text{hasManufacturer}.\{\text{Toshiba, HP}\}$	0.4
r_5	$\top$	$\exists \text{hasMarginAmount}.\{\text{High}\}$	0.01

We make a few of observations. First, all preferences except r_4 are represented by vague roles as introduced in Section 3.5.5. The reason for this is that these preferences contain non-discreteness about numbers; such as “maximize profit margin” and “quantity around 40”. The consequence is that it is not guaranteed that the notebooks with the highest score indeed have the highest possible memory (since the form of the external function defines what the user considers to be large differences in memory). Furthermore, the combination of preference scores is in our quantitative model explicit, meaning that a phrase like equally important is translated to equal score. Setting these scores will influence the final score for an order and might therefore influence the interaction with other systems. Furthermore, with the scoring approach, it is not possible to express that a preference always has a priority over another preference. For example if an order has an extremely high margin, it might still be preferred, even if it does not adhere to the preferences of Marge. We motivate this behavior by the fact that these wishes of Marge were not explicit constraints. $\square$

4.3 Traditional information retrieval and context-awareness

4.3.1 Introduction

In this section, we discuss how contextual relevance can be combined with query relevance by providing insight into the existing query-based ranking method of Berger and Lafferty (1999), who generalize the language modeling approach to information retrieval (Hiemstra, 2001; Ponte and Croft, 1998). The main reason why we chose to pursue this approach is that it not only delivers good results for traditional information retrieval but also presents a conceptually simple probabilistic model that already incorporates some hints on the influence of context data. This section results in a probability distribution that preference rules should model.

Berger and Lafferty imagine that when a user uses an information retrieval system, she starts with an information need. They imagine this information as a fragment of an “ideal document”. The user translates this ideal document fragment (in her head) into a compact query which is posed to the system. The task for the retrieval system is, given the query and a model of how the information need is translated into a query, to retrieve back the most likely (ideal) document.

Thus the task for the retrieval system becomes to determine for which document d , the probability that it is ideal, given a query q and a user u , is the highest:

$$P(d \in DocIdeal | Q=q \wedge U=u) \quad (4.1)$$

Here $d \in DocIdeal$ is the event that d is in the ideal document set, $Q=q$ is the event that query q is observed, and $U=u$ is the event that user u poses the query. By Bayes’ law, $P(d \in DocIdeal | Q=q \wedge U=u)$ can be rewritten as:

$$\frac{P(Q=q | d \in DocIdeal \wedge U=u) P(d \in DocIdeal | U=u)}{P(Q=q | U=u)} \quad (4.2)$$

Since the retrieval process should result in a ranked list of documents and the denominator, $P(Q=q | U=u)$, is for each document the same, it can be ignored for the ranking purpose, resulting in the following expression consisting of a query-dependent part and a query-independent part:

$$\underbrace{P(Q=q | d \in DocIdeal \wedge U=u)}_{\text{query-dependent}} \cdot \underbrace{P(d \in DocIdeal | U=u)}_{\text{query-independent}}$$

Berger and Lafferty suggest that the query-independent part can be used to adapt the result to the users needs and interests, but take it to be uniform for all documents. In their paper, they focus on the query-dependent part; the probability that query q is posed by the user u , given that document d is the ideal document.

4.3.2 The query relevance of a document given its features

To get more insight into the approach of Berger and Lafferty (1999), we first present their method to determine the query-dependent part:

$$P(Q=q | d \in DocIdeal \wedge U=u)$$

Berger and Lafferty assume that $DocIdeal$ is a singleton set and that the user enumerates several significant terms in her query. Ponte and Croft (1998) make two extra assumptions:

- The probability that a term t is posed by the user u , given that document d is the ideal document, is related to the frequency of the term t in the document d ; to model this, they introduce a probability distribution $P(t \in Q | d \in \text{DocIdeal} \wedge U = u)$. They call this distribution a language model.
- The probability that a user query contains a term t , given that document d is ideal is independent of the other terms in the query.

Under these assumptions, we have:

$$P(Q=q | d \in \text{DocIdeal} \wedge U=u) = \prod_{t \in q} P(t \in Q | d \in \text{DocIdeal} \wedge U=u)$$

Recall that the task for the retrieval system is to determine for which d , $P(Q=q | d \in \text{DocIdeal} \wedge U=u)$ is the highest. Given that we have the probability distribution $P(t \in Q | d \in \text{DocIdeal} \wedge U=u)$, this equation enables us to rank documents based on how likely they will address the information need of the user as expressed in the query.

4.3.3 The effects of context

Since we want to rank documents not only based on the user query but also based on the context of the user, we shall revisit Equation (4.2) to see how context plays a role in influencing the terms of the equation. Since the user was already present in the original model and we are only interested in how the context influences the information need of the user, we incorporate the context of the user in the model by considering the user as being *situated* (i.e., having a certain context). We indicate this by replacing u with s , indicating both user and her situation. Hence, the task of the information retrieval is to find the document $d \in \text{Doc}_{\text{real}}$ for which $P(d \in \text{DocIdeal} | Q=q \wedge S=s)$ is the highest. According to Bayes' law:

$$P(d \in \text{DocIdeal} | Q=q \wedge S=s) = \frac{P(Q=q | d \in \text{DocIdeal} \wedge S=s) P(d \in \text{DocIdeal} | S=s)}{P(Q=q | S=s)}$$

As can be seen, context has its effect in three places. First, it affects the denominator, implying that if a user poses a query which is less likely considering its context, the documents matching this result should be considered more relevant. Since this does not affect the ranking of the documents, we do not address it in our investigation. We imagine, however, that this term

Figure 4.2 – Both query and preference rules address properties of ideal documents.

can be used to determine the general relevance of query results in a context-aware system, for example, to determine how prominent query results should be displayed.

Second, context affects the “query generation” term $P(Q=q|d \in DocIdeal \wedge S=s)$. This term can be used to account for the various input methods under different contexts leading to a different formulation of a query, even though the “ideal document” is the same. Notwithstanding the fact that we can imagine a user posing different query terms when she uses her phone rather than her desktop PC, we assume in our investigation that the influence of context on this term can be neglected compared to the effect of context on the query-independent term.

In the previous section, it was assumed that the query-independent term, $P(d \in DocIdeal|S=s)$, was the same for each document. However, the essence of context-awareness the relevance of a document depends on the context of the user. In the remainder of this chapter, we therefore address the elaboration of this term using preference rules.

4.4 Combining scores

In Section 3.3.3 we introduced a score function σ which assigns a score to each context-aware preference rule. To combine scores we introduce in this section a so-called *context score combination operation* $\odot$ that *combines* two scores $\sigma(r)$ and $\sigma(r')$ into a score for the composite preference rule $\{r, r'\}$.

Actually, the operation comes in two flavors, namely $\otimes$ (Section 4.4.3) and $\oplus$ (Section 4.4.2), each of which is based on a particular assumption about the user’s behavior.

To be able to provide a probabilistic interpretation for the combinations,

we first describe the relation between the aggregated scores of preference rules and the ideal document for a user in Section 4.4.1.

4.4.1 Probabilistic interpretation of the aggregate score

The context-aware preference for a document in a situation, is based on the scores of the preference rules that are satisfied for this document in the situation, and the context score combination function $\odot : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$.

We assume independency between preferences (see Section 4.4.5). Similar to $\prod$ and $\sum$ being product and sum aggregates, we define the aggregate score function $\odot$ as the aggregate of $\odot$, i.e., $\odot\{a, b, \dots\} = a \odot b \odot \dots$. This requires that $\odot$ is commutative and associative. Given that the context-aware preferences for a user u are given by the set of preference rules $rs_u = \{r_1 \dots r_n\}$, we define the preference relation between two documents as follows:

$$d_0 \prec_s^{rs_u} d_1 \quad =_{def} \quad \bigodot_{r \in rs_u | r \models_s d_0} \sigma(r) < \bigodot_{r \in rs_u | r \models_s d_1} \sigma(r)$$

The only information that is available about the effect of the situation on the ideal documents is encoded in the preference rules of the user. We therefore stipulate that the probability that a document is ideal given the situation (which incorporates the user u) is equal to the probability that it is ideal given the preference rules of the user u that it satisfies.

$$P(d \in DocIdeal | S=s) = P(d \in DocIdeal | \{r \in rs_u \mid r \models_s d\} = rs) \quad (4.3)$$

As the aggregate of the scores should lead to an ordering over documents depending on their contextual information need, we require the probabilistic interpretation of the resulting score as being proportional to the probability that a document is ideal given the preference rules:

$$\bigodot_{r \in rs} \sigma(r) \propto P(d \in DocIdeal | \{r \in rs_u \mid r \models_s d\} = rs) \quad (4.4)$$

where the score σ and combinations $\odot$ are tuned to model the probability distribution.

In this section we present two combinations $\odot$ together with the consequence for the probabilistic interpretation of the score function σ . A disjunctive combination $\oplus$ is presented in Section 4.4.2, and a conjunctive combination $\otimes$ in Section 4.4.3. The reason for choosing these two is their simplicity. Although we use different symbols for the combination functions, we use

Table 4.1 – Three preference rules with their effect on the user’s preference in a situation where all rules are applicable.

(a) Preferences

Preference	$\mathcal{C}$	$\mathcal{I}$	σ
r_1	AtUniversity	Technical	0.7
r_2	AfterLunch	News	0.6
r_3	Sunny	Short	0.3

(b) Disjunctive combination

Document properties	Score
Technical, news, short	$1 - ((1 - 0.7) \cdot (1 - 0.6) \cdot (1 - 0.3)) = 0.916$
Technical, news, long	$1 - ((1 - 0.7) \cdot (1 - 0.6)) = 0.88$
Non-technical, non-news, long	$1 - 1 = 0$

(c) Conjunctive combination in case the prior probability that a document satisfies an interest of a rule is equal to the prior probability that it is not satisfied, given that the rule is applicable, and therefore $\sigma(r) = 1 - (\sigma(\bar{r}))$ (see Equation 4.13).

Document properties	Score
Technical, news, short	$0.7 \cdot 0.6 \cdot 0.3 = 0.126$
Technical, news, long	$0.7 \cdot 0.6 \cdot (1 - 0.3) = 0.294$
Non-technical, non-news, long	$(1 - 0.7) \cdot (1 - 0.6) \cdot (1 - 0.3) = 0.084$

for both interpretations the same symbol (σ) for the score function. Also note that for the disjunctive combination the score of a rule is a probability, whereas for the conjunctive combination the score of a rule is a ratio between two probabilities.

To illustrate both combinations we introduce, as an example, a user Suzanne, who has the preference rules in Table 4.1a.

Instead of a single ideal document (as imagined by Berger and Lafferty) in this thesis we consider that there is a set of documents that the user is interested in; $Doc_{Ideal} \subseteq Doc_{real}$. This set consists of all documents that satisfy the information preference of a user.

4.4.2 Disjunctive context combination

First, we introduce the disjunctive contextual preference combination. The intuition behind this combination is the assumption that contrary to, for example, keywords of a query, each preference rule defines a separate concept of interest and, in principle, a user is satisfied as soon as a document satisfies one of these concepts. This intuition is displayed in Figure 4.3.

Figure 4.3 – Situation in which the set of ideal documents is a subset of the disjunction of three interests. The total grey area is the total set of documents that is the disjunctive combination of the interests.

Figure 4.4 – Example of a simple Bayesian network, where the fact whether document d is ideal only depends on whether d is technical and the user is at the university, d is a news article and it is after lunch, and whether d is short when the weather is sunny.

Because of the intuition that a document is ideal if one of the contextual interests is satisfied, we choose to adopt a noisy-OR relation between preferences and ideality. The noisy-OR, introduced by Pearl (1988), is a generalization of the logical or, where we are allowed to have uncertainty about whether a parent causes the child to be true. This uncertainty can be seen as “noise”, which is the reason for the naming. For example in the Bayesian network in Figure 4.4 the rules are causes (parent) of the event that a document is ideal (child).

The noisy-OR makes the following two assumptions:

Accountability (all causes are listed). In our example, this means that a document is only ideal if it is technical at the university, about news after lunch, or short when it is sunny. This assumption can be relaxed when we add a so-called *leak node*, standing for all other causes that

could lead to the fact that d is ideal based on the situation.

Inhibition independence. In our example, this means that whatever causes a technical document at the university not to be ideal, is independent of whatever causes a news document after lunch not to be ideal. For example, a news article could be imperfect after lunch when Suzanne just listened to the news on the radio, and a short document could be imperfect if it is sunny, if Suzanne just got an advice from a fellow PhD student that long documents are often more relevant.

Following these assumptions, the probability that the child is false equals the aggregation of the probabilities that prevent the parents from causing the child to be true. For two parents this results in:

$$\begin{aligned} P(\neg\text{child}|\text{parent}, \text{parent}') \\ = P(\neg\text{child}|\text{parent}, \neg\text{parent}') \cdot P(\neg\text{child}|\neg\text{parent}, \text{parent}') \end{aligned}$$

this implies that, for two parents, the probability that the child is true is:

$$\begin{aligned} P(\text{child}|\text{parent}, \text{parent}') \\ = P(\text{child}|\text{parent}, \neg\text{parent}') + P(\text{child}|\neg\text{parent}, \text{parent}') \\ - P(\text{child}|\text{parent}, \neg\text{parent}') \cdot P(\text{child}|\neg\text{parent}, \text{parent}') \end{aligned}$$

and for n parents:

$$\begin{aligned} P(\text{child}|\text{parent}_1 \dots \text{parent}_n) \\ = 1 - \prod_{i:1..n} P(\neg\text{child}|\text{parent}_i \wedge \bigwedge_{x:1..(i-1),(i+1),\dots,n} \neg\text{parent}_x) \\ = 1 - \prod_{i:1..n} (1 - P(\text{child}|\text{parent}_i \wedge \bigwedge_{x:1..(i-1),(i+1),\dots,n} \neg\text{parent}_x)) \end{aligned} \quad (4.5)$$

To identify the noisy-OR relation, we take as parents the satisfiability of the rules in rs_u . The score of a rule r for a user u is then equal to the probability that a document is ideal if it only satisfies r :

$$\sigma(r) = P(d \in \text{DocIdeal} \mid \{r \in rs_u \mid r \models_s d\} = \{r\}) \quad (4.6)$$

And for a document d satisfying r and r' :

$$\begin{aligned} P(d \in \text{DocIdeal} \mid \{r \in rs_u \mid r \models_s d\} = \{r, r'\}) \\ = \text{Following noisy-OR relation} \end{aligned}$$

$$\begin{aligned}
& P(d \in \text{DocIdeal} \mid \{r \in rs_u \mid r \models_s d\} = \{r\}) \\
& + P(d \in \text{DocIdeal} \mid \{r \in rs_u \mid r \models_s d\} = \{r'\}) \\
& - P(d \in \text{DocIdeal} \mid \{r \in rs_u \mid r \models_s d\} = \{r\}) \\
& \cdot P(d \in \text{DocIdeal} \mid \{r \in rs_u \mid r \models_s d\} = \{r'\})
\end{aligned}$$

We therefore define the aggregation as follows:

$$x \oplus y = a + b - a \cdot b \quad (4.7)$$

It follows that requirement (4.4) is met:

$$\begin{aligned}
& \bigoplus_{r \in RS_u} \sigma(r) \\
& = \text{Definition aggregation, definition of } \oplus \text{ (4.7), arithmetic} \\
& \quad 1 - \prod_{r \in RS_u} (1 - \sigma(r)) \\
& = \text{Definition noisy-OR (4.5), definition of } \sigma(r) \text{ (4.6)} \\
& \quad P(d \in \text{DocIdeal} \mid \{r \in rs_u \mid r \models_s d\} = rs)
\end{aligned}$$

Note that this relation is stricter than (4.4) requires; equality instead of proportionality.

In Figure 4.3, when looking at probabilities in terms of frequencies of observed events (the so-called *frequentist view*), the score for $r = (x, \mathcal{I}_1)$ equals:

$$\sigma((x, \mathcal{I}_1)) = \frac{\#(\text{DocIdeal} \cap \mathcal{I}_1 \setminus (\mathcal{I}_2 \cup \mathcal{I}_3))}{\#(\mathcal{I}_1 \setminus (\mathcal{I}_2 \cup \mathcal{I}_3))}$$

where x is the context of the situation of Figure 4.3. To get an idea of how scores are combined, in Table 4.1b we show the disjunctive combination for the preference scores of Suzanne.

4.4.3 Conjunctive context combination

The intuition for the conjunctive contextual preference combination is the assumption that context-aware interests are more like query interests and therefore state that a document is ideal only if the document satisfies all interests of the user under this situation. This intuition is displayed in Figure 4.5.

To be able to apply conjunctive aggregation we postulate that each user has for each rule $r = (\mathcal{C}, \mathcal{I}) \in rs_u$ an inverse rule $\bar{r} = (\mathcal{C}, \neg\mathcal{I}) \in rs_u$ that is applicable in the same situations as r but satisfied by exactly those documents that are not satisfied by r . We also postulate for a set of rules rs the

Figure 4.5 – Situation in which the set of ideal documents is a subset of the conjunction of three interests. The total grey area is the total set of documents that is the conjunctive combination of the interests.

set $\bar{r}s$ that contains the inverses of the rules in rs . Therefore in the situation in which a rule r is applicable, we know that always r or $\bar{r}$ is satisfied. Furthermore, if we know that the set of satisfied rules is rs , the rules that were applicable were $rs \cup \bar{r}s$. This means that knowing that the set of rules rs is satisfied, is equal to knowing that the set of applicable rules is $rs \cup \bar{r}s$ and the interests of the rules in rs are satisfied:

$$\begin{aligned} \{r \in rs_u \mid r \models_s d\} &= rs \\ \Leftrightarrow rs &\subseteq \{r \in rs_u \mid d \in r.\mathcal{I}\} \wedge rs \cup \bar{r}s = \{r \in rs_u \mid s \in r.\mathcal{C}\} \end{aligned} \quad (4.8)$$

Similar to query interests, we can rewrite the probability that a document is ideal, given the applicable rules and the preference rules that it satisfies, $P(d \in DocIdeal \mid \{r \in rs_u \mid r \models_s d\} = rs)$, using Bayes' rule as:

$$\begin{aligned} &P(d \in DocIdeal \mid \{r \in rs_u \mid r \models_s d\} = rs) \\ &= \frac{P(\{r \in rs_u \mid r \models_s d\} = rs \mid d \in DocIdeal) \cdot P(d \in DocIdeal)}{P(\{r \in rs_u \mid r \models_s d\} = rs)} \end{aligned} \quad (4.9)$$

We consider the prior probability that a document is ideal, $P(d \in DocIdeal)$, to be equal for all documents. Therefore the probability that a document is ideal given the preference rules that it satisfies is proportional to the probability that its preference rules are satisfied by the ideal document divided by the prior probability that its preference rules are satisfied:

$$\begin{aligned} &\frac{P(\{r \in rs_u \mid r \models_s d\} = rs \mid d \in DocIdeal) \cdot P(d \in DocIdeal)}{P(\{r \in rs_u \mid r \models_s d\} = rs)} \\ &\propto \frac{P(\{r \in rs_u \mid r \models_s d\} = rs \mid d \in DocIdeal)}{P(\{r \in rs_u \mid r \models_s d\} = rs)} \end{aligned} \quad (4.10)$$

In the following deduction, we show how to identify the independent parts of the latter equation.

$$\begin{aligned}
& \frac{P(\{r \in rs_u \mid r \models_s d\} = rs \mid d \in DocIdeal)}{P(\{r \in rs_u \mid r \models_s d\} = rs)} \\
= & \text{Applying Equation (4.8)} \\
& \frac{P(rs \subseteq \{r \in rs_u \mid d \in r.\mathcal{I}\} \wedge rs \cup \bar{r}s = \{r \in rs_u \mid s \in r.\mathcal{C}\} \mid d \in DocIdeal)}{P(rs \subseteq \{r \in rs_u \mid d \in r.\mathcal{I}\} \wedge rs \cup \bar{r}s = \{r \in rs_u \mid s \in r.\mathcal{C}\})} \\
= & \text{Product rule} \\
& \frac{P(rs \subseteq \{r \in rs_u \mid d \in r.\mathcal{I}\} \wedge rs \cup \bar{r}s = \{r \in rs_u \mid s \in r.\mathcal{C}\} \mid d \in DocIdeal)}{P(rs \subseteq \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid rs \cup \bar{r}s = \{r \in rs_u \mid s \in r.\mathcal{C}\}) \cdot P(rs \cup \bar{r}s = \{r \in rs_u \mid s \in r.\mathcal{C}\})} \\
= & \text{Conditionalized product rule} \\
& \frac{P(rs \subseteq \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid rs \cup \bar{r}s = \{r \in rs_u \mid s \in r.\mathcal{C}\}, d \in DocIdeal) \cdot P(rs \cup \bar{r}s = \{r \in rs_u \mid s \in r.\mathcal{C}\} \mid d \in DocIdeal)}{P(rs \subseteq \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid rs \cup \bar{r}s = \{r \in rs_u \mid s \in r.\mathcal{C}\}) \cdot P(rs \cup \bar{r}s = \{r \in rs_u \mid s \in r.\mathcal{C}\})} \\
= & \text{Applicability of rules is independent of ideal document} \\
& \frac{P(rs \subseteq \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid rs \cup \bar{r}s = \{r \in rs_u \mid s \in r.\mathcal{C}\}, d \in DocIdeal) \cdot P(rs \cup \bar{r}s = \{r \in rs_u \mid s \in r.\mathcal{C}\})}{P(rs \subseteq \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid rs \cup \bar{r}s = \{r \in rs_u \mid s \in r.\mathcal{C}\}) \cdot P(rs \cup \bar{r}s = \{r \in rs_u \mid s \in r.\mathcal{C}\})} \\
= & \\
& \frac{P(rs \subseteq \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid rs \cup \bar{r}s = \{r \in rs_u \mid s \in r.\mathcal{C}\}, d \in DocIdeal)}{P(rs \subseteq \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid rs \cup \bar{r}s = \{r \in rs_u \mid s \in r.\mathcal{C}\})} \\
= & \\
& \frac{P(\bigwedge_{r \in rs} r \in \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid rs \cup \bar{r}s = \{r \in rs_u \mid s \in r.\mathcal{C}\}, d \in DocIdeal)}{P(\bigwedge_{r \in rs} r \in \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid rs \cup \bar{r}s = \{r \in rs_u \mid s \in r.\mathcal{C}\})} \\
= & \begin{cases} \text{Independence between satisfaction of rules given applicability,} \\ \text{and satisfaction of rule only dependent on its own} \\ \text{applicability (or its inverse)} \end{cases} \\
& \frac{\prod_{r \in rs} P(r \in \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid \{r, \bar{r}\} \subseteq \{r \in rs_u \mid s \in r.\mathcal{C}\}, d \in DocIdeal)}{\prod_{r \in rs} P(r \in \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid \{r, \bar{r}\} \subseteq \{r \in rs_u \mid s \in r.\mathcal{C}\})} \\
= & \\
& \prod_{r \in rs} \frac{P(r \in \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid \{r, \bar{r}\} \subseteq \{r \in rs_u \mid s \in r.\mathcal{C}\}, d \in DocIdeal)}{P(r \in \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid \{r, \bar{r}\} \subseteq \{r \in rs_u \mid s \in r.\mathcal{C}\})} \\
= & \text{"given } \bar{r} \text{" is the same as "given } r \text{"} \\
& \prod_{r \in rs} \frac{P(r \in \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid r \in \{r \in rs_u \mid s \in r.\mathcal{C}\}, d \in DocIdeal)}{P(r \in \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid r \in \{r \in rs_u \mid s \in r.\mathcal{C}\})}
\end{aligned}$$

To choose σ , $\oplus$, and $\bigoplus$ from Equation (4.4), we define the conjunctive score of a rule as the ratio between the probability that the rule is satisfied if it is applicable when the document is ideal and the prior probability that the rule is satisfied if it is applicable.

$$\sigma(r) = \frac{P(r \in \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid r \in \{r \in rs_u \mid s \in r.\mathcal{C}\}, d \in DocIdeal)}{P(r \in \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid r \in \{r \in rs_u \mid s \in r.\mathcal{C}\})} \quad (4.11)$$

As the resulting preference score is a multiplication of the scores of individual rules, we take

$$\otimes = \text{multiplication, and } \bigotimes_{i:0 \dots n} x_i = \prod_{i:0 \dots n} x_i \quad (4.12)$$

Let us illustrate this with an example. Suppose Suzanne has six rules $rs_u = r_1, r_2, r_3, \bar{r}_1, \bar{r}_2, \bar{r}_3$ with $r_1 = (University, Technical)$, $r_2 = (Sunny, Short)$, $r_3 = (Lunch, News)$, and therefore $\bar{r}_1 = (University, \neg Technical)$, $\bar{r}_2 = (Sunny, \neg Short)$, and $\bar{r}_3 = (Lunch, News)$. We want to know the probability that a long technical news document d is ideal for Suzanne in a situation $s = \text{“Suzanne is at the university and it is sunny”}$, $\{r \in rs_u \mid r \models_s d\} = \{r_1, \bar{r}_2\}$.

$$P(d \in DocIdeal \mid RS_{s,d} = \{r_1, \bar{r}_2\}) \\ \propto \prod_{r \in \{r_1, \bar{r}_2\}} \frac{P(r \in \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid r \in \{r \in rs_u \mid s \in r.\mathcal{C}\}, d \in DocIdeal)}{P(r \in \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid r \in \{r \in rs_u \mid s \in r.\mathcal{C}\})}$$

In words, the result is the probability that an ideal document is long and technical, given that Suzanne is at the university and it is sunny, divided by the prior probability that a document is long and technical, given that Suzanne is at the university. Notice that rule r_3 does not play a part; indeed, it is not applicable.

In the specific case that for a rule r the probability that its interest is satisfied is equal to the probability that its interest is not satisfied, given that applicability, the score of r is one minus the score of $\bar{r}$:

$$\begin{aligned} P(r \in \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid r \in \{r \in rs_u \mid s \in r.\mathcal{C}\}) \\ = P(\bar{r} \in \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid r \in \{r \in rs_u \mid s \in r.\mathcal{C}\}) \quad (4.13) \\ \implies \sigma(r) = 1 - \sigma(\bar{r}) \end{aligned}$$

In this case the system only has to store either r or $\bar{r}$ since they follow completely from each other. To get an idea of how scores are combined if this is indeed the case, in Table 4.1c we show the conjunctive combination for the preference scores of Suzanne.

4.4.4 Comparison between disjunctive and conjunctive behavior

To get an idea of the behaviors, in Figure 4.6 we plotted for a score combination of two rules the lines that present equal resulting scores. These plots show that the conjunctive behavior places relatively more emphasis on satisfying multiple rules than the disjunctive behavior. Since $\bar{x} \otimes \bar{y} = \overline{x \oplus y}$ (with $\bar{x} = 1 - x$) Figure 4.6a mirrors Figure 4.6b.

4.4.5 On the independence of preferences

For the synthesis of our score model $(\odot, \sigma)$, we assume an independence between the preferences of a user. The question is how realistic this assumption is. In general we can say that, if two preference rules address the same

(a) Conjunctive behavior for a combination of two scores: the lines $\sigma(r_1) \otimes \sigma(r_2) = x$ are plotted for various x . The larger the x , the higher the line.

(b) Disjunctive behavior for a combination of two scores: the lines $\sigma(r_1) \oplus \sigma(r_2) = x$ are plotted for various x . The larger the x , the higher the line.

Figure 4.6 – Different behavior of users with two preference rules. On each line the score combination is constant.

kind of preference the independence assumption is violated. For example, the assumption is violated if a user has two rules indicating that she prefers warm clothes when it is cold outside. In other words, two preference rules should not have the same *intention*. The user should take this into account when designing the rules by combining preferences with the same intention into a single rule.

We illustrate the consequences using an example. As combination function we use the disjunctive combination, but similar considerations hold for other combination functions. Suppose Suzanne searches for articles on XML and has the following three context-aware preference rules:

Preference	$\mathcal{C}$	$\mathcal{I}$	σ
r_1	$\top$	<i>AuthorOwnGroup</i>	0.8
r_2	$\top$	<i>AuthorOwnFloor</i>	0.7
r_3	$\top$	<i>ConferencePaper</i>	0.9

Furthermore, suppose that her group is located on the same floor. Now suppose the system has to choose between a conference paper and a technical report from Sander; someone from her group. The scores for the articles are as follows:

$$\begin{array}{lll} \text{Conference paper} & 1 - (1 - 0.9) & = 0.9 \\ \text{Report from Sander} & 1 - (1 - 0.8) \cdot (1 - 0.7) & = 0.94 \end{array}$$

One can see that the interests *AuthorOwnFloor* and *AuthorOwnGroup* have both a positive influence on the report from Sander. Furthermore, since her group is located on the same floor, the preferences will always overlap. The question is whether this violates the independence assumption between interests.

The answer depends on the *intention* the user had by preference r_1 and r_2 . If both were to express that Suzanne wanted a paper from an author nearby location, the assumption is violated. The consequence of this violation is that the nearbyness of the author might get an “unfair” advantage over conference papers (unless the user had taken this into account when assigning the scores). If, however, the intention of preference rule r_1 was different from r_2 , for example based on the fact that her group often writes very comprehensible, the interests are independent, which is the assumption of our score model.

4.5 Score acquisition from historic data

Recall from Section 3.3.1 that the acquisition of the preference (scores) can either be explicit or implicit. Following the implication that a context-aware

system should be able to automatically learn the behavior of the user and her preferences, we discuss in this section how to implicitly acquire scores from the interaction history.

If the user did not specify any preference rules for documents in contexts, there are several options at our disposal to acquire her preferences. For example, by looking at similar users (e.g., those that are nearby), and see what their preferences are and somehow take the “average” of them as the preference for the new user. A brief overview of relevant methods is given by McTear (1993).

In this section, we focus on acquiring the preferences from the interaction history of the user with a non-context-aware system. The motivation for this approach is that this information is almost always available to the system itself and there is no need for explicit preferences or a profile.

Since a situation is never completely the same, and most of the time the user will probably not be interested in *exactly* the same document as the last time when a similar context occurred, we should be careful which interest and context features to choose and be sure that they generalize over different documents and situations. For example, suppose a user watches the traffic bulletin each workday morning. In this case, we may assume that on the next workday morning, she is again interested in traffic bulletin. The user is, however, probably not interested in watching the same traffic bulletin but only in most recent bulletin. Hence, it is important that there is a feature that generalizes over different traffic bulletins.

How to determine the right kind of features is out of the scope of our research, since this is specific for each retrieval problem. In many cases, features could be provided by data suppliers (e.g., television guides, IMDB), or determined from sensor readings (e.g., localization techniques, activity recognition). Sometimes, even if we manage to choose well-generalized context and document features, the user’s behavior might vary because of influences that cannot be sensed or modeled, or just because the user sometimes wants some variation. The variation brings about the scores in our context model. Here, we focus on a behavior in which each chosen document is ideal and show how to acquire the scores in CIM; based on the disjunctive combination and based on conjunctive combination.

4.5.1 Definitions and example history

To show how to acquire the rule scores from the historic data, we introduce a notation for contents of the history. First, we suppose that the user has posed a query (e.g., “meetings in smart environments” as in Figure 4.7). For each query, the system stores in the history the situation s , the query q , the

Figure 4.7 – A typical existing search system, where a user searches for meetings in smart environments.

user of the system u , the documents that were chosen (e.g., clicked on) by the user dcs , and the documents that were looked at but not chosen $dncs$.

The system stores only specifications of documents and situations. To address the uncertainty that the (real, sensed) data exhibits, the specification of the documents and situations that the system stores depends probabilistically on the (real, sensed) data; where the probability that a certain specification is stored is equal to the probability that the real item conforms to this specification (as introduced in Section 3.5.2). For example, suppose that we only differentiate between short and long articles, and technical and non-technical articles. Suppose that the user chose one (real) document that was short with a probability of 20% and technical with a probability of 80%, then the system will choose between the following document specifications to store:

Choice	Probability that the system will store this specification
short, technical	0.16
short, not technical	0.04
not short, not technical	0.64
not short, not technical	0.16

As we use only the ratios between frequencies in the history to acquire the scores, it is easy to see that for a history that is long enough, these ratios approximate the probabilities.

This leads to the following form for a history:

$$[t = (s, q, u, dncs, dcs), \dots, t' = (s', q', u', dncs', dcs')]$$

Since we consider relevant contexts and interests to be identified beforehand (for example using their Information Gain (Yang and Pedersen, 1997)), the remaining task is to determine a score for the interests.

Another consideration is that we do not take into account the modeling of sessions over time; therefore we group all history in the following aggregates that we use in the next sections:

$$\begin{aligned}
docpres_{\mathcal{C}} &= \{t \in H, d \in (t.dcs \cup t.dncs) \mid t.s \in \mathcal{C} \bullet (t, d)\} \\
docinterest_{\mathcal{C}, \mathcal{I}} &= \{(t, d) \in docpres_{\mathcal{C}} \mid d \in \mathcal{I} \bullet (t, d)\} \\
&= \{t \in H, d \in t.dcs \cup t.dncs \mid (\mathcal{C}, \mathcal{I}) \models_{t.s} d \bullet (t, d)\} \\
docchos_{\mathcal{C}} &= \{(t, d) \in docpres_{\mathcal{C}} \mid d \in t.dcs \bullet (t, d)\} \\
&= \{t \in H, d \in t.dcs \cup t.dncs \mid t.s \in \mathcal{C} \bullet (t, d)\}
\end{aligned}$$

Thus $docpres_{\mathcal{C}}$ are those documents that are presented during situations with context $\mathcal{C}$, $docinterest_{\mathcal{C}, \mathcal{I}}$ are those documents that were presented and satisfied interest $\mathcal{I}$, and $docchos_{\mathcal{C}}$ are those documents that were chosen in situations with context $\mathcal{C}$. All documents are uniquely identified per history tuple, to count documents as relevant multiple times over the history.

In order to describe the behavior of the user, we also introduce the set $DocPresented = docpres_{\top}$ for all the documents that were presented, and $DocChosen = docchos_{\top}$ for all chosen documents.

We assume that the existing system is not context-aware, and that the probability that a preference rule is satisfied, given that it is applicable, is independent of whether the document is presented by the existing system:

$$\begin{aligned}
&P(r \in \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid r \in \{r \in rs_u \mid s \in r.\mathcal{C}\}) \\
&= P(r \in \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid r \in \{r \in rs_u \mid s \in r.\mathcal{C}\}, d \in DocPresented)
\end{aligned} \tag{4.14}$$

Of course presented documents by the system might (hopefully) have a higher probability of being ideal. We postulate that the factor with which this probability changes is independent of the satisfied preference rules. We call this factor α_{sys} .

$$\begin{aligned}
&\alpha_{sys} \cdot P(d \in DocIdeal \mid \{r \in rs_u \mid r \models_s d\} = rs) \\
&= P(d \in DocIdeal \mid \{r \in rs_u \mid r \models_s d\} = rs, d \in DocPresented)
\end{aligned} \tag{4.15}$$

which implies:

$$\begin{aligned}
&\alpha_{sys} \cdot P(d \in DocIdeal \mid r \in \{r \in rs_u \mid r \models_s d\}) \\
&= P(d \in DocIdeal \mid r \in \{r \in rs_u \mid r \models_s d\}, d \in DocPresented)
\end{aligned} \tag{4.16}$$

and:

$$\begin{aligned} & \alpha_{sys} \cdot P(d \in DocIdeal \mid r \notin \{r \in rs_u \mid r \models_s d\}) \\ &= P(d \in DocIdeal \mid r \notin \{r \in rs_u \mid r \models_s d\}, d \in DocPresented) \end{aligned} \quad (4.17)$$

From this specification we derive another expression for α_{sys} :

$$\begin{aligned} & \alpha_{sys} \\ &= \text{specification (4.15)} \\ & \quad \frac{P(d \in DocIdeal \mid \{r \in rs_u \mid r \models_s d\} = rs \wedge d \in DocPresented)}{P(d \in DocIdeal \mid \{r \in rs_u \mid r \models_s d\} = rs)} \\ &= \text{conditional probability (2}\times\text{)} \\ & \quad \frac{P(d \in DocIdeal \wedge \{r \in rs_u \mid r \models_s d\} = rs \wedge d \in DocPresented)}{P(\{r \in rs_u \mid r \models_s d\} = rs \wedge d \in DocPresented)} \cdot \frac{P(\{r \in rs_u \mid r \models_s d\} = rs)}{P(d \in DocIdeal \wedge \{r \in rs_u \mid r \models_s d\} = rs)} \\ &= \left\{ \begin{array}{l} \text{conditional probability (top-left term)} \\ \text{independence between presentation of } d \text{ and whether } rs \\ \text{are satisfied (bottom-left term).} \end{array} \right. \\ & \quad \frac{P(d \in DocPresented \mid d \in DocIdeal \wedge \{r \in rs_u \mid r \models_s d\} = rs) \cdot P(d \in DocIdeal \wedge \{r \in rs_u \mid r \models_s d\} = rs)}{P(\{r \in rs_u \mid r \models_s d\} = rs) \cdot P(d \in DocPresented)} \\ & \quad \cdot \frac{P(\{r \in rs_u \mid r \models_s d\} = rs)}{P(d \in DocIdeal \wedge \{r \in rs_u \mid r \models_s d\} = rs)} \\ &= \text{arithmetic} \\ & \quad \frac{P(d \in DocPresented \mid d \in DocIdeal \wedge \{r \in rs_u \mid r \models_s d\} = rs)}{P(d \in DocPresented)} \\ &= \left\{ \begin{array}{l} \text{independence between presentation of } d \\ \text{and whether } rs \text{ are satisfied} \end{array} \right. \\ & \quad \frac{P(d \in DocPresented \mid d \in DocIdeal)}{P(d \in DocPresented)} \end{aligned}$$

Although this gives some insight in α_{sys} , we still cannot determine it from historic data that we store.

To illustrate the effects of assumed behaviors on the scores, we present in Table 4.2 an aggregated history. For each behavior, we calculate the acquired score for the preference rule $(\mathcal{C}, \textit{Technical})$ in case there are two rules: $(\mathcal{C}, \textit{Technical})$ and $(\mathcal{C}, \textit{Short})$.

We show how to acquire the scores under the assumption that the user chooses a presented document in a context if and only if the document is ideal. That is:

$$d \in docpres_{\mathcal{C}} \wedge (d \in docchos_{\mathcal{C}} \equiv d \in DocIdeal) \quad (4.18)$$

Table 4.2 – An example of an aggregated history for a context $\mathcal{C}$. This history shows that the user chose 50 documents that were technical of which 35 were short. The total number of chosen documents is 200.

$\mathcal{I}$	$\#docinterest_{\mathcal{C},\mathcal{I}}, \#(docinterest_{\mathcal{C},\mathcal{I}} \cap docchos_{\mathcal{C}})$
$\top$	400, 200
Technical	70, 50
Short	200, 60
Technical,Short	45, 35

4.5.2 Score acquisition for the conjunctive model

We first consider the scores for the conjunctive combination, where $\odot = \otimes$. To determine the score of a preference rule r , we calculate:

$$\begin{aligned}
& \sigma(r) \\
&= \text{Definition (4.11)} \\
& \quad \frac{P(r \in \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid r \in \{r \in rs_u \mid s \in r.\mathcal{C}\}, d \in DocIdeal)}{P(r \in \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid r \in \{r \in rs_u \mid s \in r.\mathcal{C}\})} \\
&= \text{Assumption (4.14) (2}\times\text{)} \\
& \quad \frac{P(r \in \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid r \in \{r \in rs_u \mid s \in r.\mathcal{C}\}, d \in DocIdeal, d \in DocPresented)}{P(r \in \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid r \in \{r \in rs_u \mid s \in r.\mathcal{C}\}, d \in DocPresented)} \\
&= \text{Assumption (4.18)} \\
& \quad \frac{P(r \in \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid r \in \{r \in rs_u \mid s \in r.\mathcal{C}\}, d \in DocChosen)}{P(r \in \{r \in rs_u \mid d \in r.\mathcal{I}\} \mid r \in \{r \in rs_u \mid s \in r.\mathcal{C}\}, d \in DocPresented)}
\end{aligned}$$

This can be looked up the history, in the example of Table 4.2 this implies:

$$\sigma((\mathcal{C}, Technical)) = \frac{\frac{50}{200}}{\frac{70}{400}} = \frac{10}{7}$$

Note that the score of a rule can be higher than one since, for the conjunctive combination, the score is not a probability but a ratio between two probabilities.

4.5.3 Score acquisition for the disjunctive model

The score for a preference rule r when disjunctive combination of contextual interests is considered can be acquired using two methods. The first method is, similarly to the conjunctive case, to use the definition of the score for the disjunctive combination from Equation (4.6), and consider those documents that are only satisfied by $r = (\mathcal{C}, \mathcal{I})$:

$$\begin{aligned}
& \sigma(r) \\
&= \text{definition of score from Equation 4.6}
\end{aligned}$$

$$\begin{aligned}
& \sigma(r) = P(d \in \text{DocIdeal} \mid \{r \in rs_u \mid r \models_s d\} = \{r\}) \\
& = \text{system influence (4.15)} \\
& \quad \frac{1}{\alpha_{sys}} \cdot P(d \in \text{DocIdeal} \mid \{r \in rs_u \mid r \models_s d\} = \{r\}, d \in \text{DocPresented}) \\
& = \text{presented and ideal equals chosen (Equation (4.18))} \\
& \quad \frac{1}{\alpha_{sys}} \cdot P(d \in \text{DocChosen} \mid \{r \in rs_u \mid r \models_s d\} = \{r\}, d \in \text{DocPresented}) \\
& = \text{in history, } r = (\mathcal{C}, \mathcal{I}) \\
& \quad \frac{1}{\alpha_{sys}} \cdot \frac{\#(\text{DocChosen} \cap \text{docinterest}_{\mathcal{C}, \mathcal{I}} \setminus (\bigcup_{r' \in (rs_u \setminus r)} \text{docinterest}_{r', \mathcal{C}, r', \mathcal{I}}))}{\#(\text{docinterest}_{\mathcal{C}, \mathcal{I}} \setminus (\bigcup_{r' \in (rs_u \setminus r)} \text{docinterest}_{r', \mathcal{C}, r', \mathcal{I}}))}
\end{aligned}$$

The last fraction can be looked up, and although α_{sys} is still unknown, we could tune its value to the system. In the specific case that $\alpha_{sys} = 5$ in the example of Table 4.2 this implies:

$$\sigma((\mathcal{C}, \text{Technical})) = \frac{1}{5} \cdot \frac{50 - 35}{70 - 45} = \frac{3}{25}$$

An alternative approach is to look up the fraction of chosen documents for all documents that satisfy $r = (\mathcal{C}, \mathcal{I})$ to those that do not satisfy r . We calculate these fractions in turn, and call them a and b respectively:

$$\begin{aligned}
& a \\
& = \text{definition: } a = \frac{\#(\text{DocChosen} \cap \text{docinterest}_{\mathcal{C}, \mathcal{I}})}{\# \text{docinterest}_{\mathcal{C}, \mathcal{I}}}, \text{ (4.18)} \\
& \quad P(d \in \text{DocIdeal} \mid r \in \{r \in rs_u \mid r \models_s d\}, d \in \text{DocPresented}) \\
& = \text{definition (4.16)} \\
& \quad \alpha_{sys} \cdot P(d \in \text{DocIdeal} \mid r \in \{r \in rs_u \mid r \models_s d\}) \\
& \approx \text{assume: average in history is good approximation} \\
& \quad \alpha_{sys} \cdot \frac{1}{\# \text{docinterest}_{\mathcal{C}, \mathcal{I}}} \cdot \sum_{d \in \text{docinterest}_{\mathcal{C}, \mathcal{I}}} (\bigoplus_{r' \in (\{r \in rs_u \mid r \models_s d\} \setminus r)} \sigma(r') \oplus \sigma(r)) \\
& = \text{property } \frac{1}{n} \sum_{i=1}^n ((\bigoplus_{j \neq i} x_j) \oplus x) = x \oplus \frac{1}{n} \sum_{i=1}^n \bigoplus_{j \neq i} x_j \\
& \quad \alpha_{sys} \cdot (\sigma(r) \oplus \frac{1}{\# \text{docinterest}_{\mathcal{C}, \mathcal{I}}} \cdot \sum_{d \in \text{docinterest}_{\mathcal{C}, \mathcal{I}}} \bigoplus_{r' \in (\{r \in rs_u \mid r \models_s d\} \setminus r)} \sigma(r')) \\
& = \text{define } X \text{ as } ()\text{-part in preceding line} \\
& \quad \alpha_{sys} \cdot (\sigma(r) \oplus X)
\end{aligned}$$

and

$$\begin{aligned}
& b \\
& = \text{definition: } b = \frac{\#(\text{DocChosen} \setminus \text{docinterest}_{\mathcal{C}, \mathcal{I}})}{\#(\text{DocPresented} \setminus \text{docinterest}_{\mathcal{C}, \mathcal{I}})}, \text{ (4.18)} \\
& \quad P(d \in \text{DocIdeal} \mid r \notin \{r \in rs_u \mid r \models_s d\}, d \in \text{DocPresented}) \\
& = \text{definition (4.17)}
\end{aligned}$$

$$\begin{aligned}
& \alpha_{sys} \cdot P(d \in DocIdeal \mid r \notin \{r \in rs_u \mid r \models_s d\}) \\
\approx & \quad \text{assume: average in history is good approximation} \\
& \alpha_{sys} \cdot \frac{1}{\#(DocPresented \setminus docinterest_{\mathcal{C}, \mathcal{I}})} \\
& \cdot \sum_{d \in (DocPresented \setminus docinterest_{\mathcal{C}, \mathcal{I}})} \bigoplus_{r' \in (\{r \in rs_u \mid r \models_s d\} \setminus r)} \sigma(r') \\
= & \quad \text{define } X' \text{ as } ()\text{-part in preceding line} \\
& \alpha_{sys} \cdot X'
\end{aligned}$$

Now we postulate that, because of independence, the average score over the rules is equal whether r was in them or not and therefore $X = X'$, and that the approximations above are equalities. Then it follows from the above equations by simple arithmetic that:

$$\sigma(r) = \frac{a - b}{\alpha_{sys} - b}$$

In case $\alpha_{sys} = 5$ in the example of Table 4.2 this implies:

$$\sigma((\mathcal{C}, Technical)) = \frac{\frac{50}{70} - \frac{200-50}{400-70}}{5 - \frac{200-50}{400-70}} = \frac{2}{35}$$

Of course, both methods of calculating the score should provide the same answer for a user that behaves exactly according to the model. However, as a user does not often behave exactly according to a model (such as in the example of Table 4.2), the second method is preferred, since it aggregates over more combinations. We will therefore apply the second model during our experiments in Chapter 6

4.6 Implementing CIM on top of a probabilistic DBMS

In the framework from Figure 3.8, the addition of score combinations (and uncertainty) results in a probability for the applicability of the preferences.

To investigate how our ideas can be used in practice, we implemented a system that is able to rank results based on the preference rules of the user for the disjunctive model. We extended PostgreSQL with a datatype for event expressions. Furthermore, we constructed functions for creating special Description Logics views. These views contain the members of the concept, including the event of their inclusion.

First, we introduce for each rule the event $e(r)$ that the rule is not inhibited. Since in the disjunctive model, inhibition between rules is independent

(see Section 4.4.2), the probability that a document d is ideal for a user u can be written as the probability of the event that one of the rules of the user is valid and applicable, and the rule is not inhibited, formally, without proof:

$$P(d \in \text{DocIdeal}) = P\left(\bigvee_{r \in \text{rs}_u} (e(\exists_a r.\mathcal{C}(a)) \wedge e(r.\mathcal{I}(d)) \wedge e(r))\right) \quad (4.19)$$

In the system, we therefore constructed for each preference rule r a preference view that contains all documents D with for each document $d \in D$ the event $e(\exists_a r.\mathcal{C}(a)) \wedge e(r.\mathcal{I}(d)) \wedge e(r)$. For example, a preference of Suzanne for technical documents when she is at the university is created as follows:

```
SELECT * FROM
  createRuleEE($$dl_intersect(dl_oneof('('SUZANNE'))'),
  dl_exists('dr_hasLocation', dl_oneof('('UNIVERSITY'))))$$,
  $$dl_exists('dr_hasCategory',dl_oneof('('TECHNICAL'))))$$,0.7);
```

We take the (probabilistic) union of these views to provide a complete preference view for a user. This view can be queried for the documents, event expressions and probabilities as follows:

```
SELECT dl_id,dl_ee,ee_getprobindep(dl_ee) FROM preferenceview;
```

For pull queries, we took the intersection of the query result and the complete preference view to acquire the preferred tuples of the user, where the probability to which they are included in the result is a measure of their importance.

Although push queries with scores were not implemented in the prototype, it is possible to run a continuous query for documents in the preference view, and whenever a document has a certain probability for being ideal, push this document to the user.

Finally, as we discussed in Section 4.3.3, the probability that a document is ideal is also influenced by the query via term $P(Q=q|d \in \text{DocIdeal} \wedge S=s)$ in Equation 4.3:

$$P(d \in \text{DocIdeal} | Q=q \wedge S=s) = \frac{P(Q=q | d \in \text{DocIdeal} \wedge S=s) P(d \in \text{DocIdeal} | S=s)}{P(Q=q | S=s)}$$

Our model suggests to combine both influences by multiplying this probability with the probability from the preference view, $P(d \in \text{DocIdeal} | S=s)$ to determine the ideality of a document. For Boolean queries the term $P(Q=q | d \in \text{DocIdeal} \wedge S=s)$ is either zero or one, and the combination is trivial. As we focus on Boolean queries, this is not further discussed in this thesis.

4.7 Discussion

In this chapter, we have decided to compare the strength preferences by a score; $\sigma : \text{preference} \rightarrow \mathbb{R}$. For *basic* preferences the score can be acquired in various ways: it can be acquired from the historical data of a non-context-aware system, or explicitly assigned by the user, or suggested by the system on account of similar users. For the *combination* of preferences we have designed a score combination function $\odot$, coming in two flavors $\oplus, \otimes$: for a disjunctive and conjunctive behavior of the user, respectively. The design of the combination function was based on a probabilistic interpretation of ‘the score of a preference’.

Although we provided some intuition for these models, it remains to be seen whether these models are actually useful in practise. We address this question in Chapter 6, where we will show some performance results. As a consequence of the probabilistic interpretation of the preference combination, we also get an intuitive semantics for explaining adaptation, as we will see in the next chapter.

Another interesting area of future research is whether the combination of preferences can be related to the combination of evidences as done in the Dempster Shafer (DS) theory (Shafer, 1976), in which separate preferences can be modeled as basic probability assignments and probability assignments can be combined with Dempster’s rule of combination. For an initial discussion on using DS theory for modeling preferences, see our technical report (van Bunnigen and Fokkinga, 2006). An additional interesting area of future research is the relation between CIM and logical models in information retrieval, such as Situation Theory (Huibers et al., 1996).

Next to these points, another open question which should be answered in the future, is how to relate the Description Logics features to the interaction history, and the question whether more complex context-aware combinations make sense.

Explaining query answers

This chapter presents a method for providing explanations for context-aware query answers based on a justification of the ranked results. Since we assume that both the scores as well as the context uncertainty can be expressed using probabilities, the method is based on an explanation model for probabilistic databases.

5.1 Introduction

The major benefit of context-awareness is that the system guides users in their choices and thereby reduces the effort of the users; however, this comes at a price. As we saw in the previous chapters, the system can have uncertainty about both preferences and context data. Because the system works with uncertain data, the returned result may contain incorrect answers, making the system questionable and unacceptable by its end-users.

As an example, suppose an information integrator populates a relation *HSAUTHOR* (*publication*, *author*) with two tuples: (*Pub1*, *Harold*) of probability 80% and (*Pub1*, *Sander*) of probability 20%. When a user has a preference for “the papers that Harold wrote without Sander”, the probability that *Pub1* belongs to the query answer is $80\% \cdot (100 - 20\%) = 64\%$. However, if she is completely sure that *Sander* wrote *Pub1*, *Pub1* is a wrong answer.

To resolve this problem, user interaction with the system is necessary. It is desirable for the query system to open the black box and explain to the user how it arrives at a query result, and which uncertainty assumptions (probabilities) form the basis of the derived query result, so that the user can decide how much confidence to place in the answer based on her own knowledge.

In this chapter, we therefore present a method for explaining context-

aware query answers. First, we survey both general and database related research on explanation in Section 5.2. We especially focus on how explanations in the field of logic relate to our Description Logics preference rules in Section 5.2.2. From the literature we conclude that existing research mostly focuses on giving the full answer derivation process. Since it is important not to flood the user with explanations, the main contributions of this chapter are:

- a set of design principles, focusing on justification, to determine the most important parts of the full explanation (Section 5.3).
- two quality measures for the quality of explanations which are applicable for probabilistic database queries in general, namely *gain* for complete answers, and *correlation*, focusing on a single answer tuple (Section 5.4).

In the remainder of the chapter, we focus on explaining single answer tuples and therefore detail the consequences of using the *correlation* for the explanations to give in typical preference scenarios. We show how to compute an answer explanation in Section 5.5 and end the chapter with discussions in Section 5.6

5.2 Explaining answers: state of art

The notion of an explanation is very broad. Russell and Norvig (2004) generally defined an explanation of a sentence P as a set of sentences E , such that E implies P . If the sentences are known to be true, they can be used to prove P . But explanations can also include *assumptions*; sentences that would suffice to prove P if they were true. In most cases, the explanation E is preferred to be minimal, meaning that no proper subset of E is also an explanation (Russell and Norvig, 2004).

Work on explanation of answers stems from the field of logic. In the area of information systems, explanation takes two roles. Either the system provides knowledge and explanations necessary for the user to carry out her task, or alternatively the system carries out some action and then explains the need and reason for the action that the system itself has taken to the user (Johnson and Johnson, 1993). As a result, most of the explanation facilities were offered by those expert systems that used rule-based reasoning to arrive at conclusions. MYCIN (Buchanan and Shortliffe, 1984) is one of the most exemplary systems. Buchanan and Shortliffe (1984) outlined five specific reasons for an explanation from an expert system perspective,

namely, understanding, debugging, education, acceptance, and persuasion, and provided explanations by translating traces of applied rules from LISP to English. They implemented several explanation options, of which the two most important ones were: why the expert system asked the user a certain question, and how the system deduced a certain conclusion.

In the area of recommender systems, explanation mechanisms were also exploited to achieve the following seven main targets: transparency - explaining how the system works; scrutability - allowing users to tell the system it is wrong; trust - increasing the user's confidence in the system; effectiveness - helping users make good decisions; persuasiveness - convincing users to try or buy the recommended items; efficiency - helping users make decisions faster; satisfaction - increasing the ease of usability or enjoyment (Tintarev and Masthoff, 2007).

In addition, there have also been considerable studies into cognition, psychology, and philosophy of questioning and question-answering with humans and how explanation can be applied to human-computer interaction (Simmons, 1970; Lester and Porter, 1997). Most of the research in this area is concerned with language analysis of questions and generation of answers in natural language. Interesting to mention here is the work of Simitsis and Koutrika (2006) on the generation of narratives of answers to so-called *Précis* queries (Simitsis et al., 2008). These are queries over a database schema with weights (for example from a preference, see Figure 3.1). The query consists of keywords, together with a specification (in the form of a weight) of which connected relations and attributes should be included in the answer. The consequence of such a query is that the answer is in itself a multi-relation database, from which the narrative is generated.

Summarizing the previous research in this area, lots of research has been done on explanation from many viewpoints and for many purposes. It is therefore important to define an explicit purpose for our explanation, which we will do in Section 5.3. Furthermore relating to the definition of explanation of Russell and Norvig, in this chapter we do not focus on explanations of an answer in the strict sense, but rather on presenting one of the sentences in E that make up the complete explanation.

5.2.1 In database systems

In the database field, answer explanations are traditionally concerned with query plans and path selections. The purpose is mainly to tune applications to take advantage of indices, instead of explaining why an answer was returned.

The first work that considered the data provenance (or lineage) as a

suspect*	crime
Bob	Kidnapping
SUSPECT S (suspect, crime) Bob Kidnapping CRIME (type, sector) Kidnapping 1 SAWPERSON (witness, hair*, clothes*) Jane gray blue WITNESS (name, sector) Jane 1 PERSON (name, hair, clothes) Bob gray blue	
SUSPECT D (suspect*, crime) Bob Kidnapping 0.250 CRIME (type, sector) Kidnapping 1 SAWCAR (witness, car*) Jane Toyota 0.500 WITNESS (name, sector) Jane 1 DRIVES (person*, car*) Bob Toyota 0.500	
Hank	Kidnapping

Figure 5.1 – Lineage in the Trio system, where it is explained why a tuple (*Bob, Kidnapping*) is in the table ‘high suspect’; based on the fact that there was a witness for the persons appearance, and for the car that the person drove.

problem in itself was the work of Cui and Widom (2000) and Buneman et al. (2001). The latter work differentiated between the sources that influenced the answer (*why-provenance*) and the locations in source databases where the answer was extracted from (*where-provenance*).

For explaining answers under uncertainty, a first step is the translation of the work on lineage to probabilistic databases by Benjelloun et al. (2006), as implemented in the Trio system (Mutsuzaki et al., 2007). Their approach makes it possible to identify for each tuple in a query answer the complete set of source data items that produced the tuple (see for example Figure 5.1).

Nevertheless, astute readers may argue that the way of presenting all the base assumptions in the database related to resulting tuples may not be practical and user-friendly, particularly in situations where the user has (or wants) limited amount of time to examine the result, and the underlying uncertain assumptions are overwhelming. In other words, the answer explanations provided by the system must be *concise* and *vital* enough, so that users can quickly and easily justify the query answer based on their own knowledge.

Das et al. (2006) did some research in this direction, focussing on concise answers. Their contribution is a method to select the top-*m* attributes of a query. The top-*m* is identified as those attributes that best explain the

Figure 5.2 – Explanation of the minimal set of axioms from the ontology responsible for subsumption $MaleStudentWith3Daughters \sqsubseteq Parent$ from Kalyanpur et al. (2006), with irrelevant parts struck out.

ranking of an answer based on a score function. They differentiate between the top-k answers that are presented to the user and the other answers to the query that are not presented. Based on this distinction, they consider three measures of explanatory power of an attribute; its influence on the scores, its influence on the ranks (vs. the answers that are not in the top-k), and its influence on the relative rank (vs. the other answers in the top-k). The difference between their approach and ours is that they focus on the most explanatory attributes for the complete result, whereas we focus on the best explanation per item. Furthermore, we focus on a probabilistic score function that allows us to probabilistically motivate the explanation power of an item.

Finally, in the area of databases and preferences, Kießling and Köstler (2002) confirmed the need for explaining results of preferences, because preferences lead to non-explicit query results. Kießling and Köstler focus on justification by introducing functions that indicate to which degree a certain item matches the preferences.

5.2.2 Logical explanation of answers

Since we represent preferences of users in a logical framework, we first address the explanation of the answers from a logical point of view. Modeling the preference for an answer as a membership of a logical concept (and ignoring the scores), explanation should justify why a certain answer d is a member of an interest concept $\mathcal{I}$, given the ontology. Kalyanpur et al. (2006) researched the related question of how to provide explanations to justify why an ontology entails a certain statement. For example, in Figure 5.2, the justification is given why a certain ontology entails the subsumption of $MaleStudentWith3Daughters$ by $Parent$.

The exact workings of the reasoning system are out of the scope of this

Figure 5.3 – Part of the explanation for a publication that is relevant because one of the authors is a co-author. Next to the reasoning, for clarity, also the other attributes of the items under consideration are given, although they are not a necessary part of the explanation.

thesis. But for simple cases, the system could explain its reasoning by step-by-step explaining the membership of defined concepts in the interest concept $\mathcal{I}$ by their definitions from the ontology, until it arrives at stored data (that is unexplainable, since it is asserted).

For example, suppose a user *PETER* has an interest $\mathcal{I} \equiv \text{hasAuthor.CoAuthor}$ and the following statements are part of the ontology:

$$\begin{aligned}
 \text{CoAuthor} &\equiv \text{authorOf.OwnPaper} \\
 \text{OwnPaper} &\equiv \text{hasAuthor.Self} \\
 \text{Self} &\equiv \{PETER\}
 \end{aligned}$$

When the system wants to explain why it believes *PETER* prefers document *doc*, it might do so by stating that it believes that the author of *doc*, *Sander001*, is a co-author of Peter (see Figure 5.3). However, Peter might not be able to verify this information and can ask the system “Please provide an explanation why you believe that *Sander001* is a co-author.” In this case, the system shows its belief that one of the papers that Sander has written, *Jans99ctext*, is co-authored by Peter. And so forth, until there are no defined concepts left and the whole reasoning is explained.

Although explaining why a document satisfies a single interest can be relatively simple, as we add more preference rules and add the explanation of why the preference rules are applicable, based on the context, the complexity of the explanation increases quickly, as seen in Figure 5.4.

Figure 5.4 – Possible explanation tree of a document that is preferred because of two preference rules r_1 , and r_2 . In boldface the contributing beliefs are presented. For a situation the various possible fillers (*Sit-fill*) are explained.

A problem of explaining the logical reasoning why an answer is preferred is the amount of data that can contribute to the belief that answer is preferred. Furthermore, our problem is even more complex since we also have to deal with uncertainty, and elements down in the explanation tree might have more influence than the top elements (see Section 5.4.3 for some typical examples).

In the remainder of this chapter, we therefore focus on explaining only part of the assumptions, based on criteria we introduce in the next section. However, the two explanation approaches do not exclude each other (as we discuss in Section 5.6.1).

5.3 Design Principles

Building an explanation facility can benefit systems and users in various ways, as described in the related work. To fill the gap between giving only one answer probability (“non-explanation”), and giving the full answer derivation process (“over-explanation”) which was the focus of previous research, we introduce an answer explanation model for context-aware queries.

The role of explanations determines the appropriate explanation model and associated quality measurement. As context-aware systems work with uncertain assumptions to derive uncertain answers, both of which could be wrong, the explanation facility that we investigate in this study is thus aiming at *justification*. This bears most similarity to *understanding* and *debugging* from an expert system perspective, as well as *scrutability* from a recommender system perspective.

We consider that a user usually is more likely to recognize mistakes in basic assumptions leading to an answer, than mistakes in the answer itself. Therefore, by revealing possibly wrong or correct assumptions used in answer derivation, the system could let the user decide, based on her knowledge about these assumptions, how much confidence to place in the final derived context-aware answer. This explanation can also help enforce the acceptance of the system, since if the user understands why an answer is wrong, she would not question much about the behavior of the system.

For our purpose of justification, we define the following four considerations for the design of our answer explanation model for context-aware database queries.

1. *Verifiability*. In response to the justification requirement, the contents of an explanation provided must be verifiable by the users.
2. *Concision*. From a user’s viewpoint, a long list of explanation contents for an answer is normally undesirable and overwhelming, particularly

when the user has limited amount of time to examine the query result. Therefore, the explanation provided must be concise enough, standing in between the traditional no-explanation and over-explanation.

3. *Influenceability.* Closely related to the above concision requirement, the explanation contents offered must have the greatest influence upon the answer. That is, if these contents are changed, there is a high chance that the answer will be affected and changed accordingly. There is no use to explain something whose change makes no difference for the answer.
4. *Doubt.* The contents of an explanation preferably exhibit a certain degree of uncertainty. In other words, there is a relatively high chance that these contents may be wrong under certain circumstances. It makes no sense to explain something to the user, while the system is pretty sure about its correctness.

5.4 Explaining answers in a probabilistic setting

Recall from Section 4.6 in the previous chapter that the preference of a user for the disjunctive model can be seen as a probabilistic view over the set of documents. In this view, each document d is associated with a probabilistic event equal to:

$$P(P(d \in \text{DocIdeal})) = P\left(\bigvee_{r \in rs_u} (e(\exists_a r. \mathcal{C}(a)) \wedge e(r. \mathcal{I}(d)) \wedge e(r))\right) \quad (5.1)$$

Therefore, the event expression associated with a document encompasses the reasoning that lead to the document appearing in the answer, and providing the best justification for an answer to a general probabilistic database query also provides the best justification why a document is preferred. To be as general as possible we address in this chapter the general case of explaining answers to probabilistic database queries. We support our approach with examples from preference rules in Section 5.4.3, and address verifiability of higher-level answers in Section 5.4.2.

As discussed in Section 3.5.2 and Section 3.5.3, the answer to a probabilistic database query is a list of tuples originating from the base relations. Each resulting tuple comes with a probability, computed based on the probabilities of the base tuples. In a context-aware setting, uncertainty often comes from observable properties of a user, such as location, mood, written papers, etc. It is therefore that a user can more easily recognize mistakes in underlying

meaningful assumptions that lead to an answer than mistakes in the answer itself. In this study, this implies that a user is more likely to verify the truth of a base tuple or a meaningful view tuple than that of an answer.

We therefore discuss two measures that indicate the explanation quality of an assumption based on their ability to justify either a complete answer (*gain*) or a single answer tuple (*correlation*). As we are concerned with explanations for answer tuples, we focus on correlation.

In the rest of this chapter we will use $e_Q(A)$ as shorthand for the event that an answer tuple A belongs to the result of a query Q . For our preference model, this is equal to the event $A \in DocIdeal$. Furthermore, we will use $e(T)$ as shorthand for the event that an assumption represented by a tuple T is true. We call T the *assumed tuple*.

In advance of our knowledge of explanation, we introduce the notion of an explained answer.

Definition 5.4.1. *Let Q be a query. An explained answer $\mathcal{A}$ to Q is a list: $\mathcal{A} = \langle (A_1, E_1, P_1, X_1), \dots, (A_n, E_n, P_n, X_n) \rangle$, where A_i is an answer tuple, $E_i = e_Q(A_i)$ is the event expression for the derivation of A_i , $P_i = P(E_i)$ is the probability that A_i belongs to the answer to Q . In Definition 5.4.2 we define component X_i , here we only say that it is an “explanation component”. The list is ordered by decreasing P_i .*

We also postulate a function *best* that provides the answer tuple with the highest probability for a complete answer; $best(\mathcal{A})$ is the answer tuple A_i for which $P(e_Q(A_i))$ is maximal in $\mathcal{A}$; thanks to the ordering of the list $i = 1$.

Gain

For the motivation of the first measure, we imagine the following interaction process:

First, the user is presented with an explained answer $\mathcal{A}$. Then, the user provides feedback by a statement about the truth-value of the explanation X_1 for the answer tuple with the highest probability P_1 . The system then uses the feedback to recalculate a new answer, $\mathcal{A}'$, based on the new information. From this new complete answer, the user chooses the answer tuple with the highest probability: $best(\mathcal{A}')$.

The first measure we call *gain* and is defined over a complete answer $\mathcal{A}$ to a query Q . Let $\mathcal{A}'$ be the answer that the system provides in case the user states that $T = X_1$ is true, and let $\mathcal{A}''$ be the answer that the system

provides if the user states that T is false, then gain is defined as follows:

$$\begin{aligned} \text{gain}(\mathcal{A}, T) &= P(e_Q(\text{best}(\mathcal{A}'))|e(T)) \cdot P(e(T)) \\ &\quad + P(e_Q(\text{best}(\mathcal{A}''))|e(\neg T)) \cdot P(e(\neg T)) \\ &\quad - P(e_Q(\text{best}(\mathcal{A}))) \end{aligned} \quad (5.2)$$

or, in words: the gain is defined as the increase in probability of the highest ranked answer tuple for a system with an explanation feedback-process, compared to a system without this process.

If confirmation of the assumed tuple T by the user does not affect the highest ranked answer tuple i.e., $\text{best}(\mathcal{A}') = \text{best}(\mathcal{A})$, the gain can be simplified as follows:

$$\begin{aligned} \text{gain}'(\mathcal{A}, T) &= P(e_Q(\text{best}(\mathcal{A}))|e(T)) \cdot P(e(T)) \\ &\quad + P(e_Q(\text{best}(\mathcal{A}''))|e(\neg T)) \cdot P(e(\neg T)) \\ &\quad - P(e_Q(\text{best}(\mathcal{A}))) \\ &= P(e_Q(\text{best}(\mathcal{A})) \wedge e(T)) - P(e_Q(\text{best}(\mathcal{A}))) \\ &\quad + P(e_Q(\text{best}(\mathcal{A}''))|e(\neg T)) \cdot P(e(\neg T)) \\ &= -P(e_Q(\text{best}(\mathcal{A})) \wedge \neg e(T)) \\ &\quad + P(e_Q(\text{best}(\mathcal{A}''))|e(\neg T)) \cdot P(e(\neg T)) \\ &= -P(e_Q(\text{best}(\mathcal{A}))|\neg e(T)) \cdot P(e(\neg T)) \\ &\quad + P(e_Q(\text{best}(\mathcal{A}''))|e(\neg T)) \cdot P(e(\neg T)) \\ &= P(e(\neg T)) \\ &\quad \cdot (P(e_Q(\text{best}(\mathcal{A}''))|e(\neg T)) - P(e_Q(\text{best}(\mathcal{A}))|e(\neg T))) \end{aligned} \quad (5.3)$$

This situation happens in particular when answer tuples other than $\text{best}(\mathcal{A})$ are independent of T , and T is positively correlated with the best answer tuple. We will use this situation later in this chapter to relate the correlation and gain measure.

As an example of the use of gain, suppose that a system provides the following answer to a query Q for ideal documents:

$$\begin{aligned} \mathcal{A} = &\langle (d, \dots, 0.75, \text{authorOf}(\text{Sander001}, \text{Jans99ctext})), \\ &(d', \dots, 0.7, \text{hasActType}(\text{Peter}, \text{Eating})) \rangle \end{aligned}$$

Suppose also that the ideality of d' is not dependent on the explanation for d and that, in case *Sander* is not the author of *Jans99ctext*, the probability that d is ideal is smaller than the probability that d' is ideal (0.7). In this

case, the gain for the assumed tuple $authorOf(Sander001, Jans99c\text{text})$ is:

$$\begin{aligned}
& gain(\mathcal{A}, authorOf(Sander001, Jans99c\text{text})) \\
&= P(e_Q(d)|e(authorOf(Sander001, Jans99c\text{text}))) \\
&\quad \cdot P(e(authorOf(Sander001, Jans99c\text{text}))) \\
&+ P(e_Q(d')|e(\neg authorOf(Sander001, Jans99c\text{text}))) \\
&\quad \cdot P(e(\neg authorOf(Sander001, Jans99c\text{text}))) \\
&- P(e_Q(d))
\end{aligned}$$

Suppose that, given that *Sander* is in fact the author of *Jans99c\text{text}*, the probability that d is ideal is 0.9, and the probability that Sander is the author of *Jans99c\text{text}* is 0.5. This means that the possible answers after the feedback process are:

$$\begin{aligned}
\mathcal{A}' &= \langle (d, \dots, 0.9, \dots), (d', \dots, 0.7, \dots) \rangle \\
\mathcal{A}'' &= \langle (d', \dots, 0.7, \dots), (d, \dots, < 0.7, \dots) \rangle
\end{aligned}$$

The probability that the first answer tuple is correct after the feedback process is $0.5 \cdot 0.9 + 0.5 \cdot 0.7 = 0.8$. Since the original probability that the first answer tuple is correct is 0.75, the gain is $0.8 - 0.75 = 0.05$. Note that for this example we also could have used the simplified version $gain'$ instead of $gain$.

A problem of the gain is that this measure is dependent on the whole list $\mathcal{A}$. Furthermore, choosing the explanations with the highest gain can result in explanations that are not related to the first answer tuple. This is not satisfactory since our goal is to justify answer tuples in isolation.

Correlation

Following the concision and influenceability considerations, we therefore propose that the strength of the relationship between an explanation and an answer should be maximal. For example, an assumed tuple T is a perfect explanation for an answer tuple A and a query Q if $e(T) \Leftrightarrow e_Q(A)$. Hence, we choose as the explanation content the assumed tuple for which the truth-value has the maximal *correlation* with the truth-value of the answer tuple. We measure the correlation between those truth-values as the actual probability of co-occurrence between event $e_Q(A)$ and event $e(T)$ minus the probability of co-occurrence if the events were independent:

$$Corr(A, T) = P(e_Q(A) \wedge e(T)) - P(e_Q(A)) \cdot P(e(T)) \quad (5.4)$$

The correlation is most extreme when $e(T)$ and $e_Q(A)$ are either identical or opposite. If $\text{Corr}(A, T)$ is zero, then $e_Q(A)$ and $e(T)$ are independent. If $\text{Corr}(A, T)$ is larger than zero, then $e_Q(A)$ and $e(T)$ are positively correlated, meaning that an increase in the probability of either of them will also increase the probability of the other. Similarly, if $\text{Corr}(A, T)$ is smaller than zero, then they are negatively correlated. Furthermore, the maximal correlation with an answer tuple A is equal to $\text{Corr}(A, A) = P(e_Q(A)) - P(e_Q(A))^2$.

We give two other motivations for defining correlation as the quality measure of explanations. First, we write the correlation as follows:

$$\begin{aligned} \text{Corr}(A, T) = & \frac{1}{2} \cdot (P(e(T)) \cdot (P(e_Q(A)|e(T)) - P(e_Q(A))) \\ & + P(e(\neg T)) \cdot (P(e_Q(A)) - P(e_Q(A)|e(\neg T)))) \end{aligned}$$

The right hand side of this equation can be seen as the “improvement of probability” for a certain answer tuple after the explanation has been taken into account. In words: the assumed tuple is either true or false. If the assumed tuple is true, the confidence changes with

$$P(e_Q(A)|e(T)) - P(e_Q(A))$$

If the assumed tuple is false, the confidence changes with

$$P(e_Q(A)) - P(e_Q(A)|e(\neg T))$$

Since the user can verify whether the assumed tuple is true or false, the probability is improved with either the former or the latter.

Second, correlation directly addresses our considerations on doubt of the tuple and influenceability from Section 5.3, as it is the multiplication of them, as can be seen in the following deduction:

$$\begin{aligned} \text{Corr}(A, T) &= P(e_Q(A) \wedge e(T)) - P(e_Q(A))P(e(T)) \\ &= -P(e_Q(A))P(e(T)) + P(e_Q(A) \wedge e(T)) \\ &= P(e_Q(A)) - P(e_Q(A))P(e(T)) - P(e_Q(A)) \\ &\quad + P(e_Q(A) \wedge e(T)) \\ &= P(e_Q(A)) \cdot (1 - P(e(T))) \\ &\quad - (P(e_Q(A)) - P(e_Q(A) \wedge e(T))) \tag{5.5} \\ &= P(e_Q(A)) \cdot P(e(\neg T)) - P(e_Q(A) \wedge e(\neg T)) \\ &= P(e(\neg T)) \cdot (P(e_Q(A)) - \frac{P(e_Q(A) \wedge e(\neg T))}{P(e(\neg T))}) \\ &= \underbrace{P(e(\neg T))}_{\text{doubt}} \cdot \underbrace{(P(e_Q(A)) - P(e_Q(A)|e(\neg T)))}_{\text{influence}} \end{aligned}$$

Note that our notion of correlation differs from the linear correlation as measured by Pearson's correlation coefficient. In fact, our correlation refers to a degree of dependence; the predictability of one variable upon the other (in the same way as Hall (1999) describes). Usage of linear correlation in database research has been done by Brin et al. (1997) for extracting so-called correlation rules for data mining.

Relation between gain and correlation

In special circumstances gain and correlation are related as follows:

$$(a) \text{ gain}(\mathcal{A}, T) \leq \text{Corr}(\text{best}(\mathcal{A}), T)$$

$$(b) \text{ gain}(\mathcal{A}, T) = \text{Corr}(\text{best}(\mathcal{A}), T)$$

For both cases it is required, like for Equation (5.3), that a confirmation of the assumed tuple T does not affect the highest ranked tuple, i.e.,

$$(i) \text{ best}(\mathcal{A}') = \text{best}(\mathcal{A})$$

In addition to (i) a sufficient condition for (a) is that

$$(ii) \text{ There are no answer tuples negatively correlated with } e(T)$$

For (b) we have as additional sufficient condition that the probability of the highest ranked answer tuple that the system provides if the user states that T is false, is equal to the original probability of the initial highest ranked answer tuple:

$$(iii) P(e_Q(\text{best}(\mathcal{A}''))|e(\neg T)) = P(e_Q(\text{best}(\mathcal{A})))$$

Now it is easy to prove that (i) and (ii) together imply (a), and (i) and (iii) together imply (b). First, for (a):

$$\begin{aligned} & \text{gain}(\mathcal{A}, T) \\ = & \quad (i), \text{ Equation (5.3)} \\ & P(e(\neg T)) \cdot (P(e_Q(\text{best}(\mathcal{A}''))|e(\neg T)) - P(e_Q(\text{best}(\mathcal{A}))|e(\neg T))) \\ \leq & \quad (ii) - \text{so } P(e_Q(\text{best}(\mathcal{A}''))) \leq P(e_Q(\text{best}(\mathcal{A}))) \\ & P(e(\neg T)) \cdot (P(e_Q(\text{best}(\mathcal{A}))) - P(e_Q(\text{best}(\mathcal{A}))|e(\neg T))) \\ = & \quad \text{Equation 5.5} \\ & \text{Corr}(\text{best}(\mathcal{A}), T) \end{aligned}$$

Similarly, for (b):

$$\begin{aligned} & \text{gain}(\mathcal{A}, T) \\ = & \quad (i), \text{ Equation (5.3)} \end{aligned}$$

$$\begin{aligned}
& P(e(\neg T)) \cdot (P(e_Q(best(\mathcal{A}''))|e(\neg T)) - P(e_Q(best(\mathcal{A}))|e(\neg T))) \\
= & \quad \text{(iii), so } P(e_Q(best(\mathcal{A}''))|e(\neg T)) = P(e_Q(best(\mathcal{A}))) \\
& P(e(\neg T)) \cdot (P(e_Q(best(\mathcal{A}))) - P(e_Q(best(\mathcal{A}))|e(\neg T))) \\
= & \quad \text{Equation 5.5} \\
& Corr(best(\mathcal{A}), T)
\end{aligned}$$

Conclusion

Since we want to explain single answer tuples, as argued before, we choose to explain the answer tuples using the assumed tuple with the highest correlation. Therefore, in summary and reverting to definition 5.4.1, we define the answer explanation for probabilistic database queries as follows.

Definition 5.4.2. Let Q be a query and $\mathcal{A}$ an explained answer to Q , say $\mathcal{A} = \langle (A_1, E_1, P_1, X_1), \dots, (A_n, E_n, P_n, X_n) \rangle$. An **explanation component** E_i is a tuple T_i , where $T_i = \arg \max_T |Corr(A_i, T)|$

5.4.1 Examples

To illustrate the answer explanation we discuss several representative query examples. Table 5.1 is a small probabilistic database with three base relations, where the base tuples are assumed to be independent.

Example 5.4.1. A query “look for the publications where Sander and Harold were co-authors” leads to the explained answer of length 1:

Answer tuple	Complex Event	Probability	X
Pub1	$HA_P1_HA \wedge HA_P1_SA$	$0.8 \cdot 0.2 = 0.16$	$hasAuthor(Pub1, Sander)$

Intuitively, both assumed tuples indicating that Sander and Harold are authors of Pub1 are equally important for the result, since in case that either Sander or Harold is not the author of Pub1, this answer will not be returned. However, for a conjunctive expression, the most influential one which could invalidate the answer is the assumption that Sander is the author of Pub1, the most uncertain one. This is evident from the correlation which is for Harold $P(\neg HA_P1_HA) \cdot (P(e_Q(Pub1)) - P(e_Q(Pub1)|\neg HA_P1_HA)) = 0.2 \cdot (0.16 - 0.0) = 0.032$, and for Sander $P(\neg HA_P1_SA) \cdot (P(e_Q(Pub1)) - P(e_Q(Pub1)|\neg HA_P1_SA)) = 0.8 \cdot (0.16 - 0.0) = 0.128$. Therefore the system provides $X = hasAuthor(Pub1, Sander)$ as explanation for the answer tuple Pub1. $\square$

(a) Author relation

Basic Event	Pr.	HASAUTHOR (publication, author)
HA_P1_HA	0.8	(Pub1, Harold)
HA_P1_SA	0.2	(Pub1, Sander)
HA_P1_PA	0.6	(Pub1, Pavel)
HA_P2_JA	0.5	(Pub2, Jaap)
HA_P2_HE	0.8	(Pub2, Henk)

(b) Group relation

Basic Event	Pr.	HASGROUP (person, group)
GR_HA_UT	1.0	(Harold, UTwente)
GR_SA_UT	1.0	(Sander, UTwente)
GR_PA_UT	1.0	(Pavel, UTwente)
GR_JA_EI	1.0	(Jaap, Eindhoven)
GR_HE_EI	1.0	(Henk, Eindhoven)

(c) Professor relation

Basic Event	Pr.	PROF (group, professor)
PF_UT_PE	0.9	(UTwente, Peter)
PF_EI_AX	0.7	(Eindhoven, Paul)

Table 5.1 – A probabilistic database with 3 base relations. Probability values are chosen to simplify the example calculations, not to reflect a real scenario, although the structure is realistic.

This example is typical for event expressions that are a conjunction of base events. In these cases, the most uncertain event has the highest correlation. Recall that the correlation can be written as the multiplication of doubt about the assumed tuple, $P(e(\neg T))$, and influence, $P(e_Q(A)) - P(e_Q(A)|e(\neg T))$. Since the influence is for each assumed tuple the same (because $P(e_Q(A)|e(\neg T))$ is zero for all assumed tuples), a higher uncertainty about T means a higher correlation.

Example 5.4.2. A query “look for the publications of which either Sander or Harold is an author” has the explained answer:

Answer tuple	Complex Event	Probability	X
Pub1	$HA_P1_HA \vee HA_P1_SA$	$0.8 + 0.2 - (0.8 \cdot 0.2) = 0.84$	$hasAuthor(Pub1, Harold)$

The truth-value of the basic event HA_P1_HA has more influence on the result than HA_P1_SA in the disjunctive event expression, since if Harold was not the author of Pub1, it says more about the system being false than if Sander was not the author of Pub1. The correlations are:

$$\begin{aligned} Corr(Pub1, hasAuthor(Pub1, Harold)) &= 0.64 \\ Corr(Pub1, hasAuthor(Pub1, Sander)) &= 0.04 \end{aligned}$$

Therefore the system provides $X = hasAuthor(Pub1, Harold)$ as explanation for the answer tuple Pub1. $\square$

This example is typical for event expressions that are a disjunction of base events. This follows from the fact that correlation can be written as $P(e(T)) \cdot (P(e_Q(A)|e(T)) - P(e_Q(A)))$. Furthermore, because $P(e_Q(A)|e(T)) = 1$ for all assumed tuples, $P(e_Q(A)) - P(e_Q(A)|e(T))$ is the same for each assumed tuple. Therefore, a higher certainty means a higher correlation.

Example 5.4.3. A query “look for the publications of which Harold is an author but not Pavel” leads to the explained answer:

Answer tuple	Complex Event	Probability	X
Pub1	$HA_P1_HA \wedge \neg HA_P1_PA$	$0.8 \cdot (1 - 0.6) = 0.32$	$\neg hasAuthor(Pub1, Pavel)$

The correlations are:

$$\begin{aligned} Corr(Pub1, hasAuthor(Pub1, Harold)) &= 0.064 \\ Corr(Pub1, hasAuthor(Pub1, Pavel)) &= -0.112 \end{aligned}$$

The absolute value for $\text{Corr}(\text{Pub1}, \text{hasAuthor}(\text{Pub1}, \text{Pavel}))$ is the highest, so the explanation will focus on $\neg \text{hasAuthor}(\text{Pub1}, \text{Pavel})$. That is, if Pavel is not the author of Pub1, the result tends to be correct with high probability. $\square$

Example 5.4.4. A more complicated query “look for the publications of persons in the group of Professor Peter” will return the explained answer:

Answer tuple	Complex Event	Probability	X
Pub1	$PF_UT_PE \wedge$ $((HA_P1_HA \wedge GR_HA_UT) \vee$ $(HA_P1_SA \wedge GR_SA_UT) \vee$ $(HA_P1_PA \wedge GR_PA_UT))$	$0.9 \cdot (1 - 0.2 \cdot 0.8 \cdot 0.4)$ $= 0.8424$	$PROF(UTwente, Peter)$

As tuples of the *HASGROUP*-relation are certain, their correlation with the answer is zero, and they are not used as explanations in our approach. For the other relevant base tuples in the database, the correlations are:

$$\begin{aligned}
 \text{Corr}(\text{Pub1}, \text{hasAuthor}(\text{Pub1}, \text{Harold})) &= 0.04608 \\
 \text{Corr}(\text{Pub1}, \text{hasAuthor}(\text{Pub1}, \text{Sander})) &= 0.01152 \\
 \text{Corr}(\text{Pub1}, \text{hasAuthor}(\text{Pub1}, \text{Pavel})) &= 0.03456 \\
 \text{Corr}(\text{Pub1}, PROF(UTwente, Peter)) &= 0.08424
 \end{aligned}$$

So the correlation for $PROF(UTwente, Peter)$ is the highest. The system therefore explains the (only) answer tuple by the assumption that the professor of the group UTwente is Professor Peter. The interesting part of this example is that, in this case, the best explanation for an answer tuple is not one of its direct attributes. $\square$

5.4.2 Verifiable views

Apart from incorporating base tuples into an answer explanation, it is possible to consider relevant view tuples as long as these view tuples are meaningful and expressive, and thus easily verified by users. We call the views of which the membership of tuples can be verified by the user *verifiable views*. These views can be defined by the user.

For example, a user may be able to verify the high-level concept of the group that took part in a publication. This view table looks, for the example database from Table 5.1, as follows:

Figure 5.5 – A typical example of a “double fan” between publications and groups, where the truth-value of whether a professor is the head of the group can have its influence on a publication through different routes.

PUBGROUP(pub,group)	Complex Event
$(Pub1, UTwente)$	$(HA_P1_SA \wedge GR_SA_UT)$ $\vee (HA_P1_HA \wedge GR_HA_UT)$ $\vee (HA_P1_PA \wedge GR_PA_UT)$
$(Pub2, Eindhoven)$	$(HA_P2_JA \wedge GR_JA_EI)$ $\vee (HA_P2_HE \wedge GR_HE_EI)$

In case a user can indeed verify the tuples of the *PUBGROUP*-view, the explanation X for Example 5.4.4 could be *PUBGROUP*(*Pub1*, *UTwente*); the assumption that the people of *Twente* wrote *Pub1*.

The correlation between a view tuple and a result tuple is calculated in a similar way as for an ordinary tuple. To balance the influence, expressiveness, verifiability, and the associated computational complexity, it is reasonable to consider both base and view tuples as the candidate explanation components.

5.4.3 The double fan

Example 5.4.4 is a typical example of what we call the “double fan” as displayed in Figure 5.5. A double fan is a situation in which the answer tuple is dependent on a single assumed tuple via multiple ways. In Example 5.4.4, whether someone in the group of Peter wrote *Pub1*, depends via multiple authors on the assumed tuple that indicates whether *Peter* is professor of *UTwente*.

The implication of this structure is that, since there are many direct attributes (e.g., authors) that lead to a single other fact, direct attributes of the answer tuple might not have the highest correlation and be less effective as explanation than other assumptions.

Many context-aware queries can incorporate such constructions. For example:

- $\text{Group} \triangleleft \text{Student} \triangleright \text{Supervisor}$; give me groups that have students that are supervised by an ACM-fellow.
- $\text{Paper} \triangleleft \text{Author} \triangleright \text{Paper}$; give me papers of the authors that wrote a paper with me.
- $\text{Author} \triangleleft \text{Paper} \triangleright \text{Author}$; give me co-authors of an author in my group.
- $\text{Person} \triangleleft \text{Location} \triangleright \text{Person}$; give me persons who are in the same room as a friend.
- $\text{Movie} \triangleleft \text{Actors} \triangleright \text{Movie}$; give me movies of actors that played in my favorite movies.
- $\text{Movie} \triangleleft \text{Genre} \triangleright \text{Movie}$; give me movies with the same genre as my favorite movie.

5.5 Implementing the explanation system

Now that we have shown how to determine the explanation component using the correlation, we provide in Section 5.5.1 a method for efficiently computing the correlations within a probabilistic database, based on event expressions. Then we address possible optimizations in Section 5.5.2.

Since for each answer tuple in the preference view of Section 4.6 we have the event expression, we are able to extend our approach by providing an explained answer, using the basic approach, by presenting the assumed tuple with the highest correlation with each answer tuple.

5.5.1 Basic Approach

To calculate the probability of an answer tuple, one can first convert its complex event expression into an equivalent disjunctive normal form, and then apply the following inclusion-exclusion (IE) formula (e.g., see Fuhr and Rölleke (1997)):

$$\begin{aligned}
P(e_Q(A)) &= P(C_1 \vee \dots \vee C_n) \\
&= \sum_{i=1}^n (-1)^{i-1} \sum_{1 \leq j_1 < \dots < j_i \leq n} P(C_{j_1} \wedge \dots \wedge C_{j_i})
\end{aligned}$$

where $C_1 \dots C_n$ are the conjunctions of the disjunctive normal form. For each i , the formula picks all possible subsets like $\{C_{j_1}, \dots, C_{j_i}\}$ of size i , and sums their probabilities together. As base events are assumed to be independent in this study, $P(C_{j_1} \wedge \dots \wedge C_{j_i}) = P(C_{j_1}) \cdot \dots \cdot P(C_{j_i})$. At each iteration, the obtained total amount is affiliated with a factor $(-1)^{i-1}$. Finally, all the results ($i = 1, \dots, n$) are summed together to get $P(A)$.

Still, it is a challenge to compute the correlation between a base tuple and an answer tuple, since the calculation of the probability of the event that an answer tuple A belongs to the query answer (i.e., a complex event expression) has a complexity that is exponential in n , and so does the calculation of $P(e_Q(A) \wedge e(T))$ in Equation (5.4). A naive algorithm for providing correlations, that is, for each base tuple calculating $P(e_Q(A) \wedge e(T))$, would have an overall time complexity of $O(\text{uniqueEvents} \cdot c \cdot 2^n)$, where c is the average number of events in a conjunction and uniqueEvents is the number of unique base tuples in the event expression (for which the probability needs to be computed).

To reduce the overload due to result explanation, we propose to integrate the computation of the answer probability $P(e_Q(A))$ with the correlation for each relevant base tuple T . Algorithm 1 details such a calculation procedure, where A is conveyed through an event expression in a disjunctive normal form $(C_{1,1} \wedge \dots \wedge C_{1,x_1}) \vee \dots \vee (C_{n,1} \wedge \dots \wedge C_{n,x_n})$.

In the algorithm, `storage_add` is a function that adds the intermediate result `subsum` to the existing value for base tuple T in `dict`. By summing up the intermediate results for all subsets of conjunctions in which a tuple appears, we get the probability $P(e_Q(A)) - P(e_Q(A)|e(\neg T))$, which by multiplication with $P(e(\neg T))$ results in the correlation (see Equation (5.5)).

Example 5.5.1. *Suppose we query for persons that have either Sander or Pavel as a co-author for Pub1. This results (among others) in an answer with the following item:*

Answer tuple	Complex Event
Harold	$((HA_P1_HA \wedge HA_P1_SA) \vee (HA_P1_HA \wedge HA_P1_PA))$

Algorithm 1: Calculation of $P(e_Q(A))$ and $P(e_Q(A)) - P(e_Q(A)|e(\neg T))$ for each related base tuple T

input : An event expression of an answer, $e_Q(A)$, in disjunctive normal form, EE , presented to the algorithm as a set of conjunctions.

output: The probability $P(e_Q(A))$ and a dictionary storing $P(e_Q(A)) - P(e_Q(A)|e(\neg T))$ for each basic event expression (base tuple) T

```

1 total_probability = 0
2 dict =  $\emptyset$ 
3 forall ConjunctSubset  $\subseteq EE$  do
4   uniqueEvents = the set of events in ConjunctSubset
5   subsum =  $(-1)^{|ConjunctSubset|-1} \cdot \prod_{T \in \text{uniqueEvents}} P(T)$ 
6   forall  $T \in \text{uniqueEvents}$  do
7     dict = storage_add(dict,  $T$ , subsum)
8   total_probability = total_probability + subsum
9 return total_probability, dict

```

To compute the probability of this answer tuple and the tuple with the highest correlation, we present the algorithm with the set:

$$EE = \{\{HA_P1_HA, HA_P1_SA\}, \{HA_P1_HA, HA_P1_PA\}\}$$

The **ConjunctSubset** now iterates over the following subsets:

$$\begin{aligned} &\{\{HA_P1_HA, HA_P1_SA\}, \{HA_P1_HA, HA_P1_PA\}\}, \\ &\{\{HA_P1_HA, HA_P1_SA\}\}, \\ &\{\{HA_P1_HA, HA_P1_PA\}\}, \\ &\emptyset \end{aligned}$$

For each subset, **uniqueEvents** is assigned the set of unique events in the subsets:

$$\begin{aligned} &\{HA_P1_HA, HA_P1_SA, HA_P1_PA\}, \\ &\{HA_P1_HA, HA_P1_SA\}, \\ &\{HA_P1_HA, HA_P1_PA\}, \\ &\emptyset \end{aligned}$$

Next, we calculate the **subsum** for each subset:

$$\begin{aligned}
 & (-1)^1 \cdot P(HA_P1_HA) \cdot P(HA_P1_SA) \cdot P(HA_P1_PA) \\
 & (-1)^0 \cdot (HA_P1_HA) \cdot P(HA_P1_PA) \\
 & (-1)^0 \cdot (HA_P1_HA) \cdot P(HA_P1_SA) \\
 & 0
 \end{aligned}$$

which added together results in the answer probability of:

$$\begin{aligned}
 P(e_Q(\text{Harold})) &= -(0.8 \cdot 0.2 \cdot 0.6) + (0.8 \cdot 0.2) + (0.8 \cdot 0.6) \\
 &= -0.096 + 0.16 + 0.48 \\
 &= 0.544
 \end{aligned}$$

Suppose we want to calculate the correlation between the answer tuple *Harold* and the assumption that *Pavel* wrote *Pub1*. The difference between the initial probability of *Harold* having *Sander* or *Pavel* as a co-author for *Pub1*, and the probability of the answer if we knew that *Pavel* did not write *Pub1*, $P(e_Q(\text{Harold})) - P(e_Q(\text{Harold}) | \neg HA_P1_PA)$, is equal to the sum of the intermediate results for all subsets in which *Pavel* appears; $-0.096 + 0.48 = 0.384$. To calculate the correlation we multiply this with the probability that *Pavel* did not write *Pub1*:

$$\text{Corr}(e_Q(\text{Harold}), HA_P1_PA) = 0.4 \cdot 0.384 = 0.1536$$

since this is higher than the correlation for *Sander*, the system explains the answer with its assumption that *Pavel* wrote *Pub1*. $\square$

The algorithm can be easily modified to cater for the correlation between the answer tuple and an assumed view tuple, by computing for each set **uniqueEvents** also the view tuples it matches.

Although the integrated computation of the correlations reduces the complexity to $O(c \cdot 2^n)$, the enumeration of possible conjunctions in the event expression of $e_Q(A)$, via **ConjunctSubset**, still leads to an exponential time complexity.

5.5.2 Application of existing optimization techniques

The bottleneck of Algorithm 1 lies in its exponential nature in computing the probability of a complex Boolean event expression.

Figure 5.6 – Independent parts of a query

Safe query plans

To optimize, first possible *safe query plans* can be identified. Safe plans are such query plans which can guarantee the correct computation of the probability while using extensional semantics (Dalvi and Suciu, 2004b), see Section 3.5.2. A safe plan results in PTIME complexity for computing an answer probability. This also means that calculating the correlations between the answer tuple and assumed base/view tuples loses its exponential complexity.

Exploiting independency

When a safe query plan is impossible or infeasible (e.g., too ineffective compared to an unsafe plan), in the database literature, two types of optimization techniques are used: exploiting independency in sub plans (Sarma et al., 2007) and approximation (Dalvi and Suciu, 2004a; Ré et al., 2006). As this study focuses on exact answers, we exploit the use of the independency method developed by Sarma et al. (2007).

The existing method Take a complex query “*look for the publications of groups that are led by Peter*” for example. Here, we identify two independent parts of the query request, namely, which publication is by which group, and which groups are led by Peter (Figure 5.5.2).

Identifying the independent parts is non-trivial, and we point the reader to the work of Sarma et al., for an approach based on lineage. In our example, it could be determined that the underlying base-tuples of the *PUBGROUP* part and the *Professor* part are non-overlapping, and therefore independent. *Sub₁* and *Sub₂* in Figure 5.5.2 constitute the two independent parts.

Exploiting the use of independent parts, we optimize the computation (as shown by Sarma et al.). First, applying the computation upon independent

parts, means that the exponential part (e.g., the IE formula) only has to be applied to smaller subsets; e.g., the subtrees and their combinations instead of the whole tree at once. In our example, we first compute the *PUBGROUP* part (which has four disjunctions in the DNF of its event expression and thus takes 2^4 units of time), and then the *Professor* part (two disjunctions, 2^2 units of time) instead of computing them during one time (eight disjunctions, 2^8 units of time). Second, we cache the probabilities of intermediate results from the independent subtrees. For example, if the *UTwente* group also had other publications through other authors, we do not have to recompute the *Professor*-part for the resulting probability.

Applied to calculation of correlation The computation of correlation can profit similarly from the identification of independent parts. That is, correlated tuples for independent parts can be pre-computed. The key observation here is that the tuple that has the highest correlation with an intermediate result will have the highest correlation with the answer among all tuples within its sub-part.

To calculate the exact correlation between $e_Q(A)$ and the truth-value of an assumed base tuple T , we store $P(e_Q(A)|e(\neg S))$ for each intermediate result S . For example, for the intermediate result $PUBGROUP(UTwente, Pub1)$, we store $P(e(PubGroupPeter(Pub1))|e(\neg PUBGROUP(UTwente, Pub1)))$. Furthermore we store not only the intermediate probabilities $P(e(S))$, but also the most correlated assumed tuple T together with the probability $P(e(S)|e(\neg T))$.

Since the assumed tuple and the answer tuple are independent given the truth-value of the intermediate result, the correlation formula can be transformed to a formula only dependent on $P(e_Q(A))$, $P(e_Q(A)|e(\neg S))$, $P(e(S))$, $P(e(S)|e(\neg T))$, and $P(e(T))$, which are all stored in the system.

5.6 Discussion

Addressing the topic of traceability and correction, as discussed in Chapter 2, in this chapter we investigated how to provide the user with insight into the reasoning that led to a context-aware query answer. In this way, the user can decide how much confidence to place in the answer based on her own knowledge. As the reasoning takes place within a probabilistic database we generalized the challenge into building an answer explanation facility upon probabilistic databases.

We reviewed various goals of explanations in multiple areas, and designed a set of basic requirements for having answer explanations upon a probabilis-

tic database system, aiming at answer justification. Following these requirements, we introduced two measures for determining the best explanation component for a complete answer (gain) and for a single answer tuple (correlation), balancing verifiability and concision through the introduction of explainable view tuples. We integrated the computation of both an answer and its explanation, so that the extra cost incurred due to explanation is minimized.

The measures and algorithm that we proposed are applicable to probabilistic databases based on intensional semantics in general. However the practicality of the method in other domains than context-aware querying should be evaluated separately. Especially the assumption that a user is able to verify the assumptions that lead to an answer.

5.6.1 Future research

Since partial explanation of answers is a relatively new research area, there are many directions for future research.

Trade-off between verifiability and concision

For specific applications one could investigate the trade-off between verifiability and concision, for example by returning a set of assumed tuples as explanations and let the user specify the amount of explanations that she wants. However calculating this set is not trivial because a greedy approach (that is, simply taking the top- k assumed tuples with the highest correlation with the answer tuple) does not yield the set with the highest correlation.

Combining reasoning with probabilistic explanation

Another interesting area for future research is the combination of explanation of reasoning and explanation of probabilistic influence. For example, if the logical explanation process can be drawn as a tree (as in Figure 5.4), the branches can be ordered by their correlation measure. An alternative option would be that the system explains the most correlated tuple, but together with the reasoning that relates the tuple to the query answer.

Achieving more efficiency

Since the nature of probabilistic query answers is that the structure of their event expression is the same for each answer, it should be possible to use this fact for deriving more properties for correlation and gain that can lead to

more optimizations, e.g., to determine the best set of k explanations. Moreover, the event expressions for context combinations bear typical structures (see Equation (5.1)). Also, the use of estimation techniques can help to improve efficiency.

Evaluating context-aware querying

This chapter discusses how to evaluate our approach to context-aware preference querying. It discusses why evaluating complete context-aware applications is hard and presents two preliminary experiments; on acquiring preference scores from an interaction history and on the manual assignment of scores for preference rules by users.

6.1 Introduction

There is no generic measure of quality for context-awareness. Even if we focus on how context data can contribute to more knowledge of the user's information needs, the effect of context data can greatly vary depending on the application, and therefore also the best way of dealing with context data.

If the application in which our methods are used is unknown, it is difficult to evaluate them. What is the benefit of context-awareness if repeatability or consistency is broken? Is the context data used for pull or push queries; since in case of push queries different things play a role such as intrusiveness. As Abowd and Mynatt (2000) already pointed out, the evaluation of context-aware (ubiquitous) systems should be in a context as close as possible to the context of authentic use. To achieve this, one can focus on killer applications. But, as we stated in Chapter 1, there is not yet a killer application for context-awareness. At least, not to such extent that it can already benefit from the framework set out in this thesis.

Since a definite evaluation of the techniques proposed in this thesis is currently impossible, we perform preliminary tests of some testable aspects of these techniques. In this way we want to contribute to the discussion about how context-aware systems need to be evaluated (Kray et al., 2008). This is also the main contribution of this chapter.

To set the stage we first give an overview of general work on evaluating context-aware systems, together with an overview of previous research related to improving query results with (some form of) context data (Section 6.2). After that, we discuss which aspects of the framework we will evaluate experimentally in Section 6.3. This leads to two concrete experiments:

- An evaluation of the extent to which it is beneficial for the results of the system if a user manually assigns preferences. We compare, among others, both an ordering and a scoring of preferences (Section 6.4).
- An evaluation of how the disjunctive and conjunctive combination functions deal with combinations of multiple rules in practice. We acquire scores according to both the disjunctive and the conjunctive score combination functions from real and synthetically generated interaction histories. Using these scores, we evaluate the capability of the score combination functions in predicting the future behavior of the user (Section 6.5).

To place the evaluation in perspective, we give some reflections on the results of the experiments in Section 6.6.

6.2 State of art

6.2.1 Evaluation in general

Context-aware querying as we looked at it throughout this thesis falls in the domain of adaptive systems. Gena (2005) gave a general overview of methods and techniques that are useful for evaluating such systems. Many of those techniques find their origin in the field of human computer interaction (HCI). The field of HCI itself also faces new challenges for evaluation as indicated by Poppe and Rienks (2007). These challenges arise due to new sensing possibilities, diversifying physical interfaces, and a shift in initiative as well as in application purpose. Poppe and Rienks indicated that the main difficulties regarding the evaluation of these new systems are the lack of reference tasks and the lack of authentic usage contexts.

As proposed by Brusilovsky et al. (2001), Weibelzahl and Weber (2002), and Paramythis and Weibelzahl (2005), for the purpose of evaluation, adaptive systems can be decomposed into several layers. Gena described this as one of the peculiarities that differentiate the evaluation of adaptive systems from that of regular systems. We will take the structure proposed by Weibelzahl and Weber, given in Figure 6.1, as an example. An adaptive system first observes the user (1), then infers user properties (2), decides how

Figure 6.1 – Layers for evaluation of adaptive systems from Weibelzahl and Weber (2002). The dotted line represents the evaluation step as a consequence of having an understandable user representation.

to adapt (3), and finally presents the adapted interface to the user (4). Each layer builds upon the previous one, and all layers need to be evaluated to “guarantee success”. Testing with users traditionally focus on the first and fourth layers for the obvious reason that the user is directly involved in these layers.

A field known for its history of thorough evaluation is the field of information retrieval. A good overview of how to carry out an information retrieval experiment was given by Tague-Sutcliffe (1992). Within information retrieval (but also in other disciplines) there is a distinction between a so-called *operational test* and a *laboratory test*. In an operational test, one compares complete systems to each other together with their own users, databases, searchers, and search constraints. In a laboratory test, on the other hand, these sources of variability are under the control of the experimenter (Tague-Sutcliffe, 1992). Currently, the main focus in IR evaluation is on laboratory experiments. An interesting recent development is the rise of interest in the context of the search task (Ingwersen and Järvelin, 2005).

In conclusion, what we see is that there are currently many new developments in the evaluation of existing and new techniques such as context-awareness. In our approach, we focus on evaluating the properties within the user model in Figure 6.1, especially the explainability of these properties (e.g., the assumptions of the system about the user) to the user in a laboratory setting (the dotted line in Figure 6.1). In this way we do not have to deal with application specific user interfaces.

6.2.2 Context data for improving query results

A reasonable amount of research has been carried out on evaluating the effect of context-awareness on query results. We report here the kind of context

data they used, and how they acquired this data during their evaluation. All experiments reported in this section have shown positive results.

First, there are several investigations into the use of various statistics about the general workload of the underlying databases to introduce or improve ranked query results. Agrawal et al. (2003) ranked the results of database queries based on the appearance of attribute values in the database and their frequencies in the workload. They evaluated their approach on a large real estate database. To bootstrap the system, they let eight people make a total of 84 queries which they would pose if they wanted to buy a house. Five people provided test queries for which they evaluated the answers. Their research can be compared to the usage of query history and click-through data in information retrieval (e.g., Shen et al. (2005)). Another interesting approach on using database statistics together with user feedback is the work of Rode and Hiemstra (2006). Their system derived for each query a profile along the lines of time, topic, location, and genre, based on the initial query results of the retrieval system. The user was able to correct this profile, and the system took the corrected profile into account when determining the final query results. They tested their system by assessing the profiles for 50 queries of the *TREC* search topics by two users, with a clarification form shown in Figure 6.2.

Secondly, there is research on explicitly specified preferences over relations between tuples in the database. For example, a preference in the framework of Agrawal et al. (2006) was “choose Nicole Kidman over Penelope Cruz in drama movies”, and “choose Penelope Cruz over Nicole Kidman in the context of Spanish dramas”. The evaluation of Agrawal et al. focused on whether clustering this kind of preferences is possible without too much loss of information. They evaluated their approach by generating preferences using the ratings from IMDB. Koutrika and Ioannidis (2005), whose research was described in Section 3.2, addressed similar preferences. They tested the impact of their personalized answers using IMDB data, where they provided 14 users with a set of three queries to which the users themselves added two more queries that they would like to ask. They executed the queries with and without personalization and the users judged difference in results.

Finally, there is research on the impact of “real” context data, i.e., data that is outside the database. In this aspect Stefanidis et al. (2007) looked at the use of three contextual parameters for finding points of interests in Athens and Thessaloniki; accompanied people, time, and location. They evaluated their approach on 10 users. Each user was given a default profile that she could change. They used the obtained profiles to rank query results and compared the ranked results to rankings that the users made themselves. Adomavicius et al. (2005) used context data to improve tradi-

Select Subject
The documents found by your query have the following *subject profile*.
Change the suggested preferences if they do not reflect your search correctly.

Top 5 Subjects	Rank	Prefer	Dislike
Events	★★★★☆	☒	☐
Enviroment	★☆☆☆☆	☐	☐
Commerce	☆☆☆☆☆	☐	☐
Leisure	☆☆☆☆☆	☐	☐
Technology	☆☆☆☆☆	☐	☐

Deselect all for no subject preference.

Select Time
The documents found by your query have the following *time profile*.
Change the suggested time restriction if it does not reflect your search correctly.

Peak Range	Rank	Restrict to
10/1996 - 1/1997	★★★★☆	☐
1/1996 - 3/1996	★☆☆☆☆	☐
10/1997 - 2/1998	★★★★☆	☐

Deselect all for no time restriction.

Figure 6.2 – Experimental Clarification Form of Topic 363: Transportation Tunnel Disasters from Rode and Hiemstra (2006).

tional collaborative filtering. They evaluated on movie data by letting 62 students rate the movies they watched on a scale from 1-13 together with the situations in which they watched the movie: when (weekday, weekend, do not remember), where (movie theater, home, do not remember), and with whom (alone, with friends, with boyfriend/girlfriend, with family or others). The students could add their ratings through a website and could also rate movies that they had viewed in the past. Ramírez and de Vries (2006) studied the relation between context data and relevant elements of structured documents. The elements under consideration were articles, metadata (title, author, journal, year, and abstract), sections, subsections, and sub-subsections. Context features considered were: familiarity with the topics as indicated by the users on a 1-5 scale, request types into which the researchers classified the questions in terms of specificity (narrow or broad), complexity (asking about a single aspect or multiple aspects), and intentions of the users, for which the researchers classified the descriptions of the information need into *decide*, *apply*, *explain*, *study*, or *personal interest*. The experiments were done using 68 search tasks on the INEX collection; a collection of scientific articles. Finally, Wen et al. (2004) used context data for query expansion based on the workload. Their search space consists of documents containing

PC troubleshooting information, and their context consists of specific keys in the Microsoft Windows registry. They used 29 queries such as “Cannot open Outlook attachment”, where the task of the system was to provide the document explaining how to solve this problem.

Summarizing the previous work in this area, the positive outcomes of previous research indicate that the use of context data is beneficial for improving query answers. Furthermore, we see much research use existing data sets such as TREC and IMDB extended with context data. Finally, it is interesting to see that most research on using context data to improve query results has not (yet) used real sensor data as context data, but instead focuses on either imaginary situations or non-sensor data. This is understandable since the cost of creating such a context-aware data set is very high.

6.3 What to evaluate?

The hypothesis underlying our research is that the use of context data can improve query answers. Anecdotal evidence in Chapter 2 and related research surveyed in the previous section, indicated that this is in fact the case. As mentioned in Section 6.2.1, we focus on evaluating particular aspects of our approach, since the evaluation of our framework as a whole is too application specific and not yet feasible.

Addressing the topics from Chapter 2, we derived our contextual interest model CIM. First, we can evaluate whether the model itself is understandable and supports reasoning. Furthermore, we can evaluate our approach in addressing topics from Section 2.3 through this model.

Some of the choices are almost impossible to test without having real applications. For example, whether our uncertainty model with event expressions is sufficient for certain applications, or whether reasoning in Description Logics is sufficient. Therefore, we focus on the topics that, in our opinion, were both easy to test and relatively application independent, namely traceability and learning. We address these topics for the model itself and its usage.

All tests should be regarded as preliminary because of their small scale.

The model itself CIM consists of scored context-aware preference rules represented in Description Logics, together with a score combination function. We chose both the ontology-based representation language and the score combination to be understandable for the user.

That an ontology-based representation language would improve traceability for the users is not completely self-evident. Since this would heavily

depend on the user interface and the experience of the users with alternative models (e.g., pure SQL), we decided not to test this but take it as an assumption for the evaluation of this work.

What is easier to test is whether the scores and their combination are understandable for the users. We therefore assess whether it makes sense to let users assign scores to preference rules in Section 6.4. The outcome of this test is that this is in fact the case. Furthermore, the test indicates that, if users assign scores to preference rules, the system rankings are better than when the users only provide an ordering of the preference rules.

Usage of the model We verified the feasibility of the implementation by an actual implementation in PostgreSQL supporting a Description Logics based user representation with uncertainty.

Addressing the traceability of the answers, we introduced a method for providing explanations for context-aware query answers in Chapter 5. These explanations are chosen in such a way that their verification has the highest impact on the answer probability. Therefore, given that the probabilities that are stored in the database are correct, they are indeed the best explanations from a justification point of view. So there is also no need for experimental evaluation of this aspect.

Addressing the learning topic, we focus our evaluation on the learning of the scores. We therefore study the behavior of the score combination functions on synthetically generated data and focus on their performance on real data in Section 6.5. The outcome is that the scores for both score combination functions can indeed be acquired from past data using the methods that we introduced in Section 4.5. Furthermore, we get slightly better results on real data when using the conjunctive score combination function instead of the disjunctive one.

6.4 Assigning the scores by users

One of the design choices of CIM is to assign a score to each preference rule (instead of, for example, having only an order of the rules). In this section we report on our test to which extent people are able to assign their preferences in scores, in such way that a system can use these scores to rank concrete items. We also address the question whether a system can provide better rankings if users assign scores to preference rules instead of providing only an ordering of the rules.

In order to be able to evaluate the system in limited time, we use predefined rules which are applicable to certain scenarios. We present the rules in

Figure 6.3 – The evaluation process for the questionnaire.

natural language, and the task of the user is to give weights to the rules in such a way that the weights represent her preferences. An example rule is “I like to go shopping when having a bad day.” Finally, we decide to use pre-defined queries to focus on the ranking (the context-awareness part). These queries are in the scenarios of TV programs, tourist destinations, and music. An example query is “Give me possible tourist destinations in Rome.”

Thirty-two people (mostly computer science students) took part in our evaluation. For all the three scenarios, we have asked them to give weights to each rule and after that, while imagining that they were in a certain situation, rank the set of query results. An example query result is “Shopping at Via Condotti (expensive street)”. We compare their rankings to a ranking by a system that uses the presented context together with the rules. The whole evaluation process is depicted in Figure 6.3.

6.4.1 Ranking methods

The ranking results of the users were compared (using three distance measures) with six ranking results returned by the system:

The conjunctive ranking A ranking based on the conjunctive score combination function as described in Section 4.4.3. For this ranking it is assumed that the prior probability that a document satisfies an interest of a rule is equal to the prior probability that it is not satisfied, given that the rule is applicable, and therefore $\sigma(r) = 1 - (\sigma(\bar{r}))$ (see Equation 4.13).

The disjunctive ranking A ranking based on the disjunctive score combination function as described in Section 4.4.2.

An ordered ranking A ranking based on the ordering of the preference rules by their scores. The document addressed by the interest of the highest scored preference rule receives the highest rank. If the interest of this rule addresses more than one document, we look at the rule with the second-highest score, etc. This is similar to the prioritized composition of Kießling discussed in Section 4.2.2.

A collaborative ranking A ranking that does not use the preference rules but is simply the average rank over the documents of all users. Since we question a group of similar people (mostly computer science students) for exactly the same situation, we expect that the collaborative model will perform quite well.

Collaborative rankings with preferences A ranking based on the preference rules but with the addition of one rule that indicates a preference for the best-liked document on average.

For each ranking method, the scores were adapted to deal with the fact that for the conjunctive score combination function a score lower than 0.5 means aversion, but for the disjunctive one it means preference. Furthermore, we smooth the scores to lie between 0.01 and 0.99.

6.4.2 Measures

To compare the score score combination function we use three methods that indicate the similarity between the user ranking and the system ranking: a general distance measure between rankings (Kendall tau), a measure focused on the best user result (dig measure), and a measure focused on the best system result (proactiveness measure). For all measures *lower* means *better*. In this section we discuss them based on the example user- and system-ranking in Table 6.1a.

Before the measures are applied, we adjust the rankings for ties as follows. First, we remove ties from the system ranking. Since we are looking at ranking purposes in the domain of context-awareness, where the goal is to minimize user-action, we decide to place a penalty on equally ranked documents by the system, that is, treating equally ranked documents by the system to be in the worst-case order for the user (if both the user and system rankings are ties, we choose a random order for the system ranking). Second, we remove the ties from the user ranking. In case the user gives multiple documents the same ranking, we treat them to be in best-case order for the system. The rational behind this choice is that we consider that the

Table 6.1 – Example rankings

(a) Ranking of user and system

Document	User rank	System rank
d1	0	2
d2	0	3
d3	2	0
d4	3	4
d5	4	0

(b) Adjusted for ties

Document	User rank	System rank
d1	0	2
d2	1	3
d3	2	1
d4	3	4
d5	4	0

user is happy with both documents. Table 6.1b shows the result of removing the ties.

To present the measures in formula form, we introduce two bijective functions for both rankings (with ties removed). Let both rankings be of size n . The function $u : Doc \rightarrow \{0, \dots, n\}$ maps each document to a position in the user ranking, and $s : Doc \rightarrow \{0, \dots, n\}$ maps each document to a position in the system ranking.

Kendall tau distance

The (non-normalized) Kendall tau distance counts the number of pair-wise disagreements between two rankings of size n ; for the normalized version, the distance is divided by the maximal number of disagreements, $n \cdot (n - 1)/2$.

In the example table, this means that the user and system ranking have a distance of 6 and a normalized distance of 0.4 (see Table 6.2). In the normalized version, a value less than 0.5 means agreement and a value greater than 0.5 means disagreement. The non-normalized version can be interpreted as the number of swaps between adjacent documents one need to make in the system rank to go to the user rank (The Kendall tau distance is sometimes also called bubble-sort distance).

Table 6.2 – Pairwise disagreements from Table 6.1

Pair	User rank	System rank	Disagreement
d1, d2	>	>	
d1, d3	>	<	1
d1, d4	>	>	
d1, d5	>	<	1
d2, d3	>	<	1
d2, d4	>	>	
d2, d5	>	<	1
d3, d4	>	>	
d3, d5	>	<	1
d4, d5	>	<	1

In formula form:

$$K_{normalized}(u, s) = \frac{\#\{d, d' | d <_u d' \wedge d \geq_s d'\}}{n \cdot (n - 1)/2}$$

Dig measure

The second measure we use is the system rank of the document that has the highest rank of the user. This measure indicates how many answers the system has to provide to include the highest ranked document of the user. In other words: how deep the user should dig in the system results to find the best answer.

In a questionnaire given to the participating users, some people indicated willingness to get multiple answers¹. To normalize the measure, we divide it by the number of documents minus one. In formula form:

$$dig_{normalized}(u, s) = \frac{s(\arg \min_d u(d))}{n - 1}$$

For the rankings in Table 6.1, this measure would be 2, and normalized to $\frac{2}{4}$.

A related measure from information retrieval is the mean reciprocal rank (MRR), which is $1/n$, with n the position of the first ranked result of the user in the system result.

¹For documentation: there were 12 users who answered this question, on average they were willing to accept around 8 to 9 suggestions for the TV and tourist scenario.

Table 6.3 – Average normalized Kendall tau

	TV	Tourist	Music
Conjunctive	0.23	0.20	0.27
Disjunctive	0.23	0.20	0.27
Ordered	0.27	0.22	0.31
Collaborative	0.21	0.22	0.22
Conjunctive + Col	0.20	0.18	0.22
Disjunctive + Col	0.21	0.18	0.22

Proactiveness measure

The last measure we use is the user’s ranking of the document that has the highest rank of the system. This measure indicates how satisfied the user is if the system proactively presents its best-ranked document. The questionnaire that we hold indicates that people are often willing to accept their second or third ranking².

Again we can normalize the result by dividing it by the number of documents minus one. In formula form:

$$proactive_{normalized}(u, s) = \frac{u(\arg \min_d s(d))}{n - 1}$$

For the rankings in Table 6.1, this measure would be 4, and normalized to $\frac{4}{4}$.

6.4.3 Results

In Table 6.3, we find the adapted normalized Kendall tau distance for all scenarios. The collaborative filtering method is the best single model on all scenarios. Furthermore, the methods that make use of the scoring perform better than the methods that only use the ordering of the preferences (statistically significant for Kendall tau in TV and music scenario). Finally, ranking methods incorporating a simple version of collaborative filtering as an extra preference rule seems to often improve collaborative filtering (although not statistically significant).

In Table 6.4, we find the average normalized dig measure for all scenarios and ranking methods. The results show a similar pattern. The results of

²For documentation: 12 users who chose to add acceptable answers for the tourist scenario on average chose 3.5 acceptable answers of the 6 presented answers, 28 users who chose to add acceptable answers for the TV scenario chose on average 3.4 acceptable answers of the 10 presented answers. 21 users who chose to add acceptable answers for the music scenario chose on average 2.5 acceptable answers of the 6 presented answers.

Table 6.4 – Average normalized dig measure

	TV	Tourist	Music
Conjunctive	0.23	0.14	0.26
Disjunctive	0.23	0.14	0.26
Ordered	0.27	0.19	0.31
Collaborative	0.19	0.13	0.18
Conjunctive + Col	0.16	0.11	0.20
Disjunctive + Col	0.17	0.11	0.20

Table 6.5 – Average normalized proactive measure.

	TV	Tourist	Music
Conjunctive	0.22	0.11	0.23
Disjunctive	0.22	0.11	0.21
Ordered	0.28	0.19	0.26
Collaborative	0.19	0.13	0.23
Conjunctive + Col	0.16	0.07	0.20
Disjunctive + Col	0.18	0.07	0.23

Table 6.5 are interesting since in these results, the combination of preference rules and collaborative filtering (Conjunctive + Col) outperforms all other methods (although not statistically significant).

Additional survey questions result in the outcome that many users see the benefit of adaptation and many users indicate that they would be prepared to fill in a profile and update it if it would lead to better query answers. Some people do not want adaptation; they want their results to be reproducible. Finally, people explicitly indicate that they only would fill in their preferences if theirs are not shared or that their privacy is guaranteed.

6.5 Using the disjunctive and conjunctive score combination functions as classifiers

Recall that the scores of preference rules indicate their importance. The disjunctive and conjunctive score combination functions define the preference of the user when multiple preference rules are applicable at the same time. They are based on different user behaviors and imply a different interpretation of the scores.

In this section, we report on the evaluation of both score combination functions by using them as classifiers that are trained on the interaction

history to predict how the user will respond to new documents. For this we compare them with the Naive Bayes classifier; a simple linear classifier that we introduce in Section 6.5.1. After the introduction of Naive Bayes we describe our evaluation method (Section 6.5.2). Then we test, on artificial data, whether the score combination functions are capable of predicting the preference of a user in case the user behaves according to their assumed behavior (Section 6.5.3). After that, we compare the score score combination functions on real data of movie ratings in Section 6.5.4.

For the conjunctive classifier, it should be noted that it will contain twice as many rules as the disjunctive classifier, as we do not require that $\sigma(r) = 1 - \sigma(\bar{r})$. However it might be possible that even if this requirement is fulfilled, we could have the same performance (with a different score acquisition method). The point of this section is to show that acquisition is possible and giving a lower boundary on the performance (in terms of having best results and the smallest model). For the conjunctive classifier, we choose $\alpha_{sys} = 1$, i.e., we assume that the underlying system does not influence the presentation of ideal documents.

6.5.1 The Naive Bayes classifier

Although we introduced the disjunctive and conjunctive score score combination functions because of their semantics, using them as classifiers calls for a comparison to existing classifiers. Therefore we compare the performance of the score combination functions with the performance of a classifier with one of the simplest internal models that is available: the Naive Bayes classifier. Classifiers with more complex models might *overfit* history data and thereby being unable to generalize and predict future preferences.

To determine the probability of the class given its features, $P(C=c \mid F_1=f_1 \wedge \dots \wedge F_n=f_n)$, for the Naive bayes classifier, Bayes' rule is used to reformulate the probability as:

$$\begin{aligned} P(C=c \mid F_1=f_1 \wedge \dots \wedge F_n=f_n) \\ = \frac{P(F_1=f_1 \wedge \dots \wedge F_n=f_n \mid C=c) \cdot P(C=c)}{P(F_1=f_1 \wedge \dots \wedge F_n=f_n)} \end{aligned} \quad (6.1)$$

The term $P(F_1=f_1 \wedge \dots \wedge F_n=f_n)$ does not depend on the class and can therefore be ignored when determining the most probable class. The “naive” assumption of Naive Bayes is that each feature is conditionally independent on every other feature, and therefore:

$$\begin{aligned} P(F_1=f_1 \wedge \dots \wedge F_n=f_n \mid C=c) \\ = P(F_1=f_1 \mid C=c) \cdot \dots \cdot P(F_n=f_n \mid C=c) \end{aligned} \quad (6.2)$$

The Naive Bayes classifier determines, from the history, for all classes and features the conditional probability of a feature, given a class. For the classification it chooses the most probable class given the features:

$$\text{classify}(f_1 \dots f_n) = \arg \max_c P(C=c) \prod_{i=1} P(F_i=f_i \mid C=c) \quad (6.3)$$

In our case, there are two classes: ideal and non-ideal, and the features are the rules that are satisfied.

6.5.2 Evaluation method

For each test we compare the performance of the disjunctive score combination function, the conjunctive score combination function, and a Naive Bayes classifier.

The data of all tests consists of a set of documents identified by their attributes, together with a “judgment” for each document whether the user chose it. For each test, we split the set of documents into a *test set* and a *training set* and acquire the scores (according to the score combination functions) from the training set as described in Section 4.5. We use the test set to determine the performance. During the test we simulate the situation that the user, given her preferences, asks for the best document that is available. For each classifier we can get a prediction (in the form of a score) per document in the test set for whether it is ideal. We rank the documents based on these scores to obtain for each classifier a list of ranked documents.

In information retrieval, performance of binary results is normally measured using *precision* (the fraction of retrieved documents that is relevant), and *recall* (the fraction of relevant documents that is retrieved). As context-aware systems should reduce the effort of the user, it is important that the quality of the highest ranked results improves. Therefore, we will report the precision for the five documents that are ranked the highest averaged over all queries, the so-called *precision at 5* (P@5). A higher precision at 5 means a better classifier. The highest possible precision is 1.

To illustrate the significance of the performance differences, we report results of the *sign test* for the *average precision* (AP). The average precision for an information need q is defined as the average of the precision measure for the top documents, cut off after each relevant document. If the set of relevant documents for an information need q is $rel(q)$ and $R_{q,d} \subseteq rel(q)$ is

the set of ranked results for q from the top result until document d , then:

$$AP(q) = \frac{1}{|rel(q)|} \sum_{d \in rel(q)} Precision(R_{q,d})$$

The sign test is used to test the significance of an improvement of one method A over another method B . This is done by disproving a null hypothesis H_0 that states that there is no difference between method A and B .

We define that method A is better than method B for an information need q , $A \succ_q B$, if its average precision is higher for q :

$$A \succ_q B \equiv AP_A(q) > AP_B(q) \quad (6.4)$$

We report the results of the sign test using $A > B$, $A < B$, $A \gg B$, and $A \ll B$. For these results $A \gg B$ means that for a set of information needs, there are more information needs for which A is better and, even, that the probability that this difference in performance was observed if, in fact, both methods were equally good for each information need, is smaller than 1%. Similarly for $A > B$, with a percentage of 5%.

6.5.3 On synthetically generated data

In this section we study how the classifiers perform in case a user *does* exhibit the behavior on which the score combination functions are founded. The expectation is that the classifiers are able to capture the behavior and will perform well under these circumstances.

Data sets

We generated two data sets both consisting of 1000 assessed documents with 15 attributes for 600 users. In the first data set, all users exhibit conjunctive behavior, and in the second data set they all exhibit disjunctive behavior. Furthermore, we assume that there are no external influences on the choices of the users.

For the disjunctive data set, we create four rules per user, all with a different interest $\mathcal{I}$ equal to a single attribute. The probability that a document satisfies an interest is 20%. The scores of the rules are 0.7, 0.7, 0.5, and 0.5. We have chosen these scores in such a way that users choose a reasonable amount of documents. The average amount of documents that they choose is equal to the probability that a document is ideal, which is around 40%.

For the conjunctive data set we also create four rules per user. For this data set we assume that for each rule the prior probability that a document

Table 6.6 – Evaluation results on synthetically generated data, with the effect of noise.

(a) Precision at 5

Behav.	noise	P@5 conj.	P@5 disj.	P@5 NB	P@5 Rand.
Conj.	0	1.0000	1.0000	1.0000	0.1467
..	0.01	0.9910	0.9903	0.9913	0.1527
..	0.1	0.8803	0.8783	0.8807	0.2263
Disj.	0	0.9417	0.9430	0.9423	0.3933
..	0.01	0.9417	0.9453	0.9450	0.4073
..	0.1	0.8530	0.8580	0.8577	0.4330

(b) Sign tests (<, >: 5%, <<, >>: 1%, see text)

Behav.	noise	conj. vs disj.	conj. vs NB	disj. vs NB
Conj.	0	>>	>>	<<
..	0.01	>>	>>	<<
..	0.2	>>	>>	<<
Disj.	0	<<	<<	>>
..	0.01	<<	<<	>>
..	0.2	<<	<<	>>

satisfies an interest of this rule is equal to the prior probability that it is not satisfied, given that the rule is applicable, and therefore $\sigma(r) = 1 - \sigma(\bar{r})$ (see Equation 4.13). However, since in the conjunctive case a score lower than 0.5 means aversion instead of preference, the scores are adapted to be in the range $[0.5, 1.0]$ (and their inverses in $[0.0, 0.5]$), leading to the scores 0.85, 0.85, 0.75, and 0.75 respectively. We choose a prior probability that a document is ideal of 0.5, and the prior probability that a document satisfies an interest is also 0.5. Finally we choose the probability that a document is chosen to be linearly correlated with the probability that it is ideal. For both disjunctive and conjunctive data set we switch the choice of the user for a certain percentage of the documents, to represent noise.

Results

To get some more insight into the effect of the noise on the performance of the classifier, we vary the amount of noise as seen in Table 6.6.

The tests reveal that all classifiers are able to capture conjunctive and disjunctive behavior of the user if it is present in the data. Furthermore, when a user behaves according to the underlying behavior of a score combination

function, this score combination function performs better in terms of average precision. Interestingly, the Naive Bayes classifier has a performance for the average precision that always lies between the performances of the two score combination functions. Finally, as expected, noise has a negative influence on the performance of all classifiers.

6.5.4 On real data

Data set

The real data used is the MovieLens data (GroupLens , 2008). It consists of approximately one million ratings for 3900 movies by 6040 users. The reason for choosing this data set is that it is a large data set that presents preferences of real users over items together with a set of clear attributes for those items. We first pre-process the data to include only ratings for users that have 100 or more ratings. This leaves us 1473 users with 302516 ratings. The average number of ratings per user is 205, the median is 171. We split the ratings in positive ratings (4 or 5) and neutral or negative ratings (1-3). This gives us 164965 positive and 137551 neutral or negative ratings. For each user, the ratings are stored for each movie together with the genres of the movie. In total, there are 18 different genres and on average a movie has (a fraction more than) two genres. For determining the significance, each user is equal to a single information need q .

Results

Table 6.7 reports the results of this test. It also shows the results for a classifier that assigns a random preference for each document in the test set. All classifiers perform better than this classifier, (statistically significant, not in Table 6.7b).

From these results we conclude that for this data set, using the conjunctive score combination function gives better predictions about the interest of the users than using either the Naive Bayes classifier or the disjunctive score combination function. This suggests that, if there are no other considerations, the conjunctive score combination function is to be preferred for predicting the interest of users over the disjunctive score combination function and the Naive Bayes classifier. Although the improvement in performance is for low training percentages statistically significant, looking at the difference in precision, the difference between the three classifiers is so small that when there are reasons to choose one of the other classifiers, these reasons will most likely be more important than this small difference in performance.

Table 6.7 – Evaluation results on real data

(a) Precision at 5

% trainingset	P@5 conj.	P@5 disj.	P@5 NB	P@5 rand
30	0.6716	0.6546	0.6349	0.5604
50	0.6675	0.6544	0.6527	0.5595
70	0.6758	0.6616	0.6570	0.5595

(b) Sign tests (<, >: 5%, <<, >>: 1%, see text)

% trainingset	conj. vs disj.	conj. vs NB	disj. vs NB
30	>>	>>	>>
50	>>	0	0
70	0	0	>

6.6 Reflections, lessons learnt

The main conclusion from this chapter is that evaluating context-aware querying is hard due to its application dependence and the high cost of creating context-aware data sets. Our main contribution to the discussion regarding the evaluation of ubiquitous systems is in line with the layered approach for evaluating adaptive systems from Weibelzahl and Weber (2002), namely to evaluate specific aspects of the complete system separately. The main novelty is that we focus on understandable user representations.

For the experiments we focused on the scoring and score combination for preference rules. First, we looked at the feasibility of assigning scores to preferences by users in Section 6.4. This particular evaluation indicated that assigning scores was useful for ranking concrete items. Another outcome (Section 6.4.3) was that assigning scores to rules contains more (useful) information than only ordering the rules. Although it seems logical, the fact that scoring of rules actually gives better rankings than ordering is a nice result. Finally, collaborative filtering almost always outperforms the non-collaborative methods. Although our test is very favorable towards collaborative filtering, it does indicate that collaborative filtering is a technique that should benefit context-aware systems. It is interesting that incorporating collaborative filtering as a preference rule seems fruitful. This observation partly indicates how to proceed with the previous observation, namely to let preference rules and collaborative filtering co-exist.

In addition, to test whether the score score combination functions capture the behavior of the user, we used them as classifiers to predict future behavior

of users. Tests on artificial data reveal that all classifiers are able to capture the behavior of the user if it is present in the data, and when a user behaves according to the behavior underlying the score combination functions, the score combination function in question performs better. Furthermore, the conjunctive score combination function slightly outperformed the disjunctive score combination function as well as a Naive Bayes classifier on real data. We should however make the remark here, that users might exhibit different behavior on different data sets, so our results might be different for other applications.

6.6.1 Future research

Applicability of results in different environments

In this chapter we used the score score combination function to predict the behavior of users for rating movies. It is possible that the behavior of users is different in other domains. It is interesting to compare the behavior in combining preferences by users in different areas.

Combining collaborative measures and preference rules

One of the interesting outcomes of letting the users assign scores to their preferences is that incorporating a preference rule based on the ratings of other users, showed the best results. We expect that more research in incorporating personal preferences with preferences of peers might result in even better results.

Evaluation leading to prescriptive models

In this thesis we tried to pick up some aspects of context-awareness focused on explainability, arguing that this is one of the application independent aspects of context-awareness. More research in the evaluation of these aspects might result in theoretical (pre-scriptive) models. An example of such a model in Human Computer Interaction (HCI) is GOMS; a theory on how long it takes a normal user to move a mouse, press a button, etc. (John and Kieras, 1996). Research in this direction for context-awareness might for example lead to a measure of privacy versus engagement or satisfaction.

Conclusions

7.1 Our achievement

Nowadays, more and more data becomes available in digital form. To be able to guide users through this wealth of data, a possibility is to adapt the provided data to the current situation (i.e., context) of the user. In this way, the newly available data itself is used to take care of its caused information-overload.

To support applications that use context data to guide the user, we proceeded in this thesis from a framework for context-aware applications founded upon database querying. Through context-aware adaptation of the answers for certain queries to the underlying database system, applications can achieve context-awareness in an application-independent way.

Context data, however, has some specific characteristics compared to ordinary data. We therefore posed the following research question:

- Which data management topics need to be addressed to incorporate context data into database querying?

The answer to this question followed from the fact that context data is often acquired from sensors and often used in a proactive manner. We identified a set of four topics that we explicitly addressed in this thesis: (1) Traceability and correction, (2) Uncertainty, (3) Learning and reasoning, and (4) Storage and logging of context data. It is our belief that other data management topics discussed in Chapter 2, such as distributiveness of the data sources, can be addressed orthogonally.

The topics that mostly influence our approach were how to deal with the uncertainty of context data and, following the uncertainty, how to provide the user with insights into the working of the system, thereby supporting traceability and correction.

Before a system can use context data to guide a user, it should know how context data influences the needs of the user. We formulated this requirement in the following research question:

- How can the effect of context data on the information need of the user be stored in a user representation?

As an answer to this question, the major contribution of this thesis is the proposal of the contextual information interest model CIM. This model stores the effect of context data on information need of the user in a set of weighted preference rules, together with a score combination function. Each rule indicates an interest of the user, under a specific context. The interests and contexts are represented in Description Logics, to support reasoning mechanisms and keep the rules understandable for the users. The score combination function determines the strength of the user's preference in case a document is satisfied by multiple preference rules at the same time. Two score combination functions are provided that attach to the scores a probabilistic interpretation and follow a conjunctive and disjunctive behavior.

We evaluated the understandability of the model with respect to the scores of preference rules. This particular evaluation indicated that assigning scores by users was in fact useful for ranking concrete items. Another outcome was that assigned scores contain more (useful) information than only an ordering of the rules.

Proceeding from this model, we addressed the topics resulting from the first research question.

We showed how to implement the model on top of an existing DBMS, to be able to easily store and log context data. Uncertainty is addressed using probabilistic event expressions that were implemented in the prototype implementation by creating a new data-type for event expressions. Having a probabilistic interpretation of the scores pays off here, since in this case, the score combination for the disjunctive combination can be done within the event expression model.

We showed how to acquire the scores of preferences by looking at the choices of the user in the past. For both the disjunctive and conjunctive score combination functions, a relation between scores and the choices of the user is given. Tests showed that, if the conjunctive or disjunctive behavior is present in the data, the behavior can be mined from the data. Furthermore, we were able to model the preferences of real users, using their history, in such a way that a user representation could predict the preference for unseen documents better than random.

The last research question addressed the use of context data.

- How can the representation of the user together with the context data lead to context-aware query answers?

Because CIM has a direct translation to an existing database, the application of our model can provide a set of preferred documents in the database (together with a score). We showed how the system can either push the most preferred document(s) proactively to the user, or re-rank existing query results, based on the preference of the user.

For these query answers, the topics of traceability, addressing uncertainty, and the probabilistic interpretation of scores of preference rules come together. By addressing both uncertainty of context data and scoring through event expressions (a form of lineage), we showed that it is possible to present the user with insight into the reasoning that led to a context-aware query answer. In this way, the user can decide how much confidence to place in the answer based on more knowledge.

Returning to our central question:

- How can context data be used to support database querying?

In this thesis, we showed how context data, taken into account its specific characteristics, can be used to provide proactive answers and rank existing query answers based on their relevance to the users, via a user representation.

However, there are two important points that, when addressed, could really complete a convincing argument for our method. The first one is the evaluation of our complete method in practice within real large-scale applications. Following this first point, the second point is to address within such applications also the other data management topics from Chapter 2. We discuss these points (and others) in more detail, in the next section.

7.2 Future research

As our research is exploratory, the reader is probably left with more questions than answers. In this section, we list our most prominent questions for future research.

Development of evaluation methods Evaluating approaches to context-awareness is hard, since on the application-side, there are not yet concrete tasks on which different systems can be compared. The development of such representative tasks is difficult since the situation of the user is essential for context-aware systems and capturing realistic scenarios will cost much time

and money. However, such tasks are necessary for the comparison of different approaches to context-awareness.

Specific to this thesis, it would be beneficial to have some kind of application environment, in which our complete approach to context-awareness can be tested, for example, in terms of user satisfaction or task performance of users.

Addressing the remaining data management topics of using context data

This thesis discussed an important set of data management topics that need to be addressed to incorporate context data into database querying. However, there are still some topics left to address, such as the distributiveness of the sensors. Although we expect that they can be addressed orthogonally, it is interesting to study how these topics relate to the methods proposed in this thesis. For example, it might be beneficial to integrate the reasoning with distributiveness to obtain more efficient reasoning.

“Native” database support for primitives The score combination model and the uncertainty in our model are both addressed by implementing an extra data type in PostgreSQL for event expressions. Since this solution is quite generic, it might pay off to have native support for score models in the database, thereby also addressing information retrieval challenges.

On a broader level, it is our opinion that database systems should manage data in an application-independent way. As more and more data is available, there are more and more applications that require the same type of data management support, such as support for scores, reasoning, or explanation. Therefore, it might pay off to research native database support of more primitives, thereby profiting from existing database research and possible benefits of integrating these new primitives.

Bibliography

Gregory D. Abowd and Elizabeth Mynatt. Charting past, present, and future research in ubiquitous computing. *ACM Trans. Comput.-Hum. Interact.*, 7(1):29–58, 2000. ISSN 1073-0516.

Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.*, 23(1):103–145, 2005. ISSN 1046-8188.

Rakesh Agrawal, Ralf Rantza, and Evimaria Terzi. Context-sensitive ranking. In Chaudhuri et al. (2006), pages 383–394. ISBN 1-59593-256-9.

Rakesh Agrawal and Edward L. Wimmers. A framework for expressing and combining preferences. In *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 297–306, New York, NY, USA, 2000. ACM Press.

Sanjay Agrawal, Surajit Chaudhuri, Gautam Das, and Aristides Gionis. Automated ranking of database query results. In *CIDR*, 2003.

Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, August 2002. ISSN 0163-6804.

Stavros Antifakos, Adrian Schwaninger, and Bernt Schiele. Evaluating the effects of displaying uncertainty in context-aware applications. In Nigel Davies, Elizabeth Mynatt, and Itiro Siio, editors, *UbiComp 2004: Ubiquitous Computing: 6th International Conference*, pages 54–69. Springer-Verlag Heidelberg, 2004.

Alessandro Artale, Enrico Franconi, Nicola Guarino, and Luca Pazzi. Part-whole relations in object-centered systems: An overview. *Data Knowledge Engineering*, 20(3):347–383, 1996.

Franz Baader, Diego Calvanese, D.L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook : theory, implementation, and applications*. Cambridge University Press, 2003a. ISBN 0-521-78176-0.

Franz Baader, R. Küsters, and Frank Wolter. Extensions to description logics. In Franz Baader, Diego Calvanese, D.L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, *The Description Logic Handbook*, pages 219–261. Cambridge University Press, 2003b. ISBN 0-521-78176-0.

Franz Baader and Werner Nutt. Basic description logics. In Franz Baader, Diego Calvanese, D.L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, *The Description Logic Handbook*, pages 1–47. Cambridge University Press, 2003. ISBN 0521781760.

Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–16, New York, NY, USA, 2002. ACM. ISBN 1-58113-507-6.

Omar Benjelloun, Anish Das Sarma, Alon Y. Halevy, and Jennifer Widom. Uldbs: Databases with uncertainty and lineage. In Umeshwar Dayal, Kyu-Young Whang, David B. Lomet, Gustavo Alonso, G.M. Lohman, Martin L. Kersten, Sang Kyun Cha, and Young-Kuk Kim, editors, *VLDB*, pages 953–964. ACM, 2006. ISBN 1-59593-385-9.

Adam L. Berger and John D. Lafferty. Information retrieval as statistical translation. In *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 15-19, 1999, Berkeley, CA, USA*, pages 222–229. ACM, 1999.

Philippe Bonnet, Praveen Seshadri, and Johannes Gehrke. Towards sensor database systems. In *2nd International Conference on Mobile Data Management*, pages 3–14, 2001.

Alexander Borgida. Description logics in data management. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):671–682, 1995. ISSN 1041-4347.

Alexander Borgida and Ronald J. Brachman. Loading data into description reasoners. In *SIGMOD '93: Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 217–226, New York, NY, USA, 1993. ACM Press.

Alexander Borgida and Ronald J. Brachman. Conceptual modeling with description logics. In *The Description Logic Handbook*, pages 349–372. Cambridge University Press, 2003. ISBN 0-521-78176-0.

Alexander Borgida, Maurizio Lenzerini, and Riccardo Rosati. Description logics for databases. In *The Description Logic Handbook*, pages 462–484. Cambridge University Press, 2003. ISBN 0-521-78176-0.

Stephan Börzsönyi, Donald Kossmann, and Konrad Stocker. The skyline operator. In *Proceedings of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, Germany*, pages 421–430. IEEE Computer Society, 2001.

Stephane Bressan, Kofi Fynn, Cheng Hian Goh, Stuart E. Madnick, Tito Pena, and Michael D. Siegel. Overview of a prolog implementation of the context interchange mediator. In *International Conference and Exhibition on The Practical Applications of Prolog*, pages 83–93, 1997.

Sergey Brin, Rajeev Motwani, and Craig Silverstein. Beyond market baskets: Generalizing association rules to correlations. In Joan Peckham, editor, *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*, pages 265–276. ACM Press, 1997.

Peter J. Brown, Winslow Burleson, Mik Lamming, Odd-Wiking Rahlff, Guy Romano, Jean Scholtz, and Dave Snowdon. Context-awareness: Some compelling applications. In *CH12000 Workshop on The What, Who, Where, When, Why and How of Context-Awareness*, April 2000.

P. Brusilovsky, C. Karagiannidis, and D. Sampson. The benefits of layered evaluation of adaptive applications and services. *Empirical Evaluation of Adaptive Systems. Proceedings of workshop at the Eighth International Conference on User Modeling, UM2001*, pages 1–8, 2001.

Bruce G. Buchanan and Edward H. Shortliffe. *Rule Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project (The Addison-Wesley series in artificial intelligence)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984. ISBN 0201101726.

Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan. Why and where: A characterization of data provenance. In *Database Theory - ICDT 2001: 8th International Conference, London, UK, January 2001. Proceedings*, volume 1973 of *LNCS*, page 316, 2001.

Matthew Chalmers. A historical view of context. *Computer Supported Cooperative Work (CSCW), The Journal of Collaborative Computing (to appear)*, 0, 2004. ISSN 0925-9724.

S. Chatterjea, T. Nieberg, Y. Zhang, and P. J. M. Havinga. Energy-efficient data acquisition using a distributed and self-organizing scheduling algorithm for wireless sensor networks. In J. Aspnes, C. Scheideler, A. Arora, and S. Madden, editors, *Proceedings of the Third IEEE Conference on Distributed Computing in Sensor Systems, DCOSS 2007, Santa Fe, USA*, volume 4549 of *Lecture Notes in Computer Science*, pages 368–385, Berlin, June 2007. Springer Verlag. ISBN 3-540-73089-3.

Surajit Chaudhuri, Gautam Das, Vagelis Hristidis, and Gerhard Weikum. Probabilistic ranking of database query results. In Mario A. Nascimento, M. Tamer Özsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer, editors, *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*, pages 888–899. Morgan Kaufmann, 2004.

Surajit Chaudhuri, Vagelis Hristidis, and Neoklis Polyzotis, editors. *Proceedings of the ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, USA, June 27-29, 2006*, 2006. ACM. ISBN 1-59593-256-9.

Guanling Chen and David Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, 2000.

Harry Chen, Tim Finin, and Anupam Joshi. Semantic web in a pervasive context-aware architecture. *Artificial Intelligence in Mobile System*, pages 33–40, 2003.

Harry Chen, Tim Finin, and Anupam Joshi. The soupa ontology for pervasive computing. In *Ontologies for Agents: Theory and Experiences*. Springer, (Whitestein Series in Software Agent Technologies), June 2005. ISBN 3-7643-7237-0.

Mitch Cherniack, Michael J. Franklin, and Stanley B. Zdonik. Data management for pervasive computing. Tutorial at VLDB, September 2001.

Jan Chomicki. Preference formulas in relational queries. *ACM Trans. Database Syst.*, 28(4):427–466, 2003.

CiteSeer (2008). Citeseer. <http://citeseer.ist.psu.edu/>, 2008.

Patricia Dockhorn Costa. *Architectural Support for Context-Aware Applications: From Context Models to Services Platforms*. PhD thesis, University of Twente, December 2007.

Yingwei Cui and Jennifer Widom. Lineage tracing in a data warehousing system. In *ICDE*, pages 683–684, 2000.

Nilesh N. Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. In Mario A. Nascimento, M. Tamer Özsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer, editors, *VLDB*, pages 864–875. Morgan Kaufmann, 2004a.

Nilesh N. Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. In Mario A. Nascimento, M. Tamer Özsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer, editors, *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*, pages 864–875. Morgan Kaufmann, 2004b.

Gautam Das, Vagelis Hristidis, Nishant Kapoor, and S. Sudarshan. Ordering the attributes of query results. In Chaudhuri et al. (2006), pages 395–406. ISBN 1-59593-256-9.

A. de Keijzer. *Management of Uncertain Data - towards unattended integration*. PhD thesis, Univ. of Twente, Enschede, February 2008.

Amol Deshpande, Carlos Guestrin, Samuel R. Madden, Joseph M. Hellerstein, and Wei Hong. Model-driven data acquisition in sensor networks. In *VLDB 2004*, pages 588–599, 2004.

Richard W. DeVaul and Alex Pentland. The ektara architecture: The right framework for context-aware wearable and ubiquitous computing applications., 2000.

Anind K. Dey. Understanding and using context. *Personal Ubiquitous Comput.*, 5(1):4–7, 2001. ISSN 1617-4909.

Anind K. Dey and Gregory D. Abowd. Towards a better understanding of context and context-awareness. Technical Report GIT-GVU-99-22, Georgia Institute of Technology, 1999.

Anind K. Dey, Gregory D. Abowd, and Jennifer Mankoff. Distributed mediation of imperfectly sensed context in aware environments. Technical Report GIT-GVU-00-14, Georgia Institute of Technology, September 2000.

Anind K. Dey, Gregory D. Abowd, and Daniel Salber. A context-based infrastructure for smart environments. In *1 st Intl. Workshop on Managing Interactions in Smart Environments (MANSE'99)*, 1999.

Anind K. Dey and Jennifer Mankoff. Designing mediation for context-aware applications. *ACM Trans. Comput.-Hum. Interact.*, 12(1):53–80, 2005.

Anind K. Dey, Timothy Sohn, Sara Streng, and Justin Kodama. icap: Interactive prototyping of context-aware applications. In Kenneth P. Fishkin, Bernt Schiele, Paddy Nixon, and Aaron J. Quigley, editors, *Pervasive*, volume 3968 of *Lecture Notes in Computer Science*, pages 254–271. Springer, 2006. ISBN 3-540-33894-2.

Paul Dourish. What we talk about when we talk about context. *Personal and Ubiquitous Computing*, 8(1):19–30, 2004. ISSN 1617-4909.

ECHISE(2005). Echise. first international workshop on exploiting context histories in smart environments, collocated with pervasive 2005, May 2005.

Ling Feng, Peter M. G. Apers, and W. Jonker. Towards context-aware data management for ambient intelligence. In *Conference on Database and Expert Systems Applications*, pages 422–431, 2004.

Norbert Fuhr and Thomas Rölleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Trans. Inf. Syst.*, 15(1):32–66, 1997.

Fabien L. Gandon and Norman M. Sadeh. Semantic web technologies to reconcile privacy and context awareness. *Web Semantics Journal*, 1(3):241–260, 2004.

Cristina Gena. Methods and techniques for the evaluation of user-adaptive systems. *Knowl. Eng. Rev.*, 20(1):1–37, 2005. ISSN 0269-8889.

Rosalba Giugno and Thomas Lukasiewicz. P-shoq(d): A probabilistic extension of shoq(d) for probabilistic ontologies in the semantic web. In Sergio Flesca, Sergio Greco, Nicola Leone, and Giovambattista Ianni, editors, *Logics in Artificial Intelligence, European Conference, JELIA 2002, Cosenza, Italy, September, 23-26, Proceedings*, pages 86–97. Springer, 2002.

- Google Scholar (2008). Google scholar. <http://scholar.google.com/>, 2008.
- Kevin Goslar and Alexander Schill. Modeling contextual information using active data structures. In *Workshop for Pervasive Information Management*, pages 325–334, 2004.
- Philip D. Gray and Daniel Salber. Modelling and using sensed context information in the design of interactive applications. In *Proceedings of the 8th IFIP International Conference on Engineering for Human-Computer Interaction*, pages 317–335. Springer-Verlag, 2001.
- GroupLens (2008). Grouplens. <http://www.grouplens.org/>, 2008.
- Tao Gu, Hung Keng Pung, and Da Qing Zhang. A bayesian approach for dealing with uncertain contexts. In *Second International Conference on Pervasive Computing (Pervasive 2004)*, April 2004.
- M. A. Hall. *Correlation-based Feature Selection for Machine Learning*. PhD thesis, The University of Waikato, 1999.
- Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, and Paul Webster. The anatomy of a context-aware application. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom)*, pages 59–68. ACM Press, 1999.
- Karen Henriksen and Jadwiga Indulska. Modelling and using imperfect context information. In *Second IEEE International Conference on Pervasive Computing and Communications. Workshop on Context Modelling and Reasoning (CoMoRea'04)*, pages 33–37. IEEE Computer Society, 2004.
- Karen Henriksen, Jadwiga Indulska, and Andry Rakotonirainy. Modeling context information in pervasive computing systems. In *First International Conference on Pervasive Computing*, pages 167–180, 2002.
- Djoerd Hiemstra. *Using Language Models for Information Retrieval*. PhD thesis, PhD thesis, University of Twente, 2001.
- Stefan Holland and Werner Kießling. Situated preferences and preference repositories for personalized database applications. In *23rd International Conference on Conceptual Modeling (ER 2004)*, pages 511–523. Springer-Verlag, November 2004.
- Ian Horrocks, Lei Li, Daniele Turi, and Sean Bechhofer. The instance store: D1 reasoning with large numbers of individuals. In Volker Haarslev and

Ralf Möller, editors, *Proceedings of the 2004 International Workshop on Description Logics (DL2004)*, Whistler, British Columbia, Canada, June 6-8, 2004. CEUR-WS.org, 2004.

Ian Horrocks and Peter F. Patel-Schneider. A proposal for an owl rules language. In Stuart I. Feldman, Mike Uretsky, Marc Najork, and Craig E. Wills, editors, *Proceedings of the 13th international conference on World Wide Web, WWW 2004, New York, NY, USA, May 17-20, 2004*, pages 723–731. ACM, 2004.

Eric Horvitz, Jack S. Breese, David Heckerman, David Hovel, and Koos Rommelse. The lumière project: Bayesian user modeling for inferring the goals and needs of software users. In Gregory F. Cooper and Serafin Moral, editors, *UAI*, pages 256–265. Morgan Kaufmann, 1998.

T. W. C. Huibers, M. Lalmas, and C.J. van Rijsbergen. Information retrieval and situation theory. *SIGIR Forum*, 30(1):11–25, 1996. ISSN 0163-5840.

Peter Ingwersen and Kalervo Järvelin. *The Turn: Integration of Information Seeking and Retrieval in Context (The Information Retrieval Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005. ISBN 140203850X.

Manfred Jaeger. Probabilistic reasoning in terminological logics. In *KR*, pages 305–316, 1994.

Bonnie E. John and David E. Kieras. The goms family of user interface analysis techniques: comparison and contrast. *ACM Trans. Comput.-Hum. Interact.*, 3(4):320–351, 1996. ISSN 1073-0516.

Hilary Johnson and Peter Johnson. Explanation facilities and interactive systems. In *IUI '93: Proceedings of the 1st international conference on Intelligent user interfaces*, pages 159–166, New York, NY, USA, 1993. ACM. ISBN 0-89791-556-9.

Gareth J. F. Jones and Peter J. Brown. Context-aware retrieval for ubiquitous computing environments. In *Mobile HCI Workshop on Mobile and Ubiquitous Information Access*, pages 227–243. Springer, September 2004.

A. Kalyanpur, B. Parsia, B. C. Grau, and E. Sirin. Justifications for entailments in expressive description logics. Technical report, University of Maryland Institute for Advanced Computer Studies (UMIACS), 2006.

Werner Kießling. Foundations of preferences in database systems. In *VLDB*, pages 311–322, 2002.

Werner Kießling. Preference queries with sv-semantics. In Jayant R. Haritsa and T.M. Vijayaraman, editors, *Advances in Data Management 2005, Proceedings of the Eleventh International Conference on Management of Data, January 6, 7, and 8, 2005, Goa, India*, pages 15–26. Computer Society of India, 2005.

Werner Kießling, Bernd Hafenrichter, Stefan Fischer, and Stefan Holland. Preference xpath: A query language for e-commerce. In *Proc. 5th Intern. Konf. für Wirtschaftsinformatik*, pages 425–440, September 2001.

Werner Kießling and Gerhard Köstler. Preference sql - design, implementation, experiences. In *VLDB*, pages 990–1001, Hong Kong, China, 2002.

Tim Kindberg, Abigail Sellen, and Erik Geelhoed. Security and trust in mobile interactions: A study of users perceptions and reasoning. In Nigel Davies, Elizabeth Mynatt, and Itiro Siio, editors, *UbiComp 2004: Ubiquitous Computing: 6th International Conference*, pages 196–213. Springer-Verlag Heidelberg, September 2004.

Alfred Kobsa and Wolfgang Pohl. The user modeling shell system bgp-ms. *User Model. User-Adapt. Interact.*, 4(2):59–106, June 1994.

Kimberle Koile, Konrad Tollmar, David Demirdjian, Howard Shrobe, and Trevor Darell. Activity zones for context-aware computing. In *Fifth International Conference on Ubiquitous Computing (UbiComp'03)*, pages 90–106, 2003.

Daphne Koller, Alon Y. Levy, and Avi Pfeffer. P-classic: A tractable probabilistic description logic. In *AAAI/IAAI*, pages 390–397, 1997.

Panu Korpipää, Miika Koskinen, Johannes Peltola, Satu-Marja Mäkelä, and Tapio Seppänen. Bayesian approach to sensor-based context awareness. *Personal Ubiquitous Computing*, 7(2):113–124, 2003a. ISSN 1617-4909.

Panu Korpipää, Jani Mäntyjärvi, Juha Kela, Heikki Keränen, and Esko-Juhani Malm. Managing context information in mobile devices. *IEEE Pervasive Computing*, 2(3):42–51, 2003b.

Georgia Koutrika and Yannis E. Ioannidis. Personalized queries under a generalized preference model. In *Proceedings of the 21st International Conference on Data Engineering, ICDE 2005, 5-8 April 2005, Tokyo, Japan*, pages 841–852. IEEE Computer Society, 2005.

Christian Kray, Lars Bo Larsen, Patrick Olivier, Margit Biemans, Arthur Hugo van Bunningen, Mirko Fetter, Tim Jay, Vassilis-Javed Khan, Gerhard Leitner, Ingrid Mulder, Jorg Muller, Thomas Plotz, and Irene Lopez de Vallejo. Evaluating ubiquitous systems with users (workshop proceedings). In *Workshop Proceedings of AmI 2007*, 2008. to appear.

M. Lacroix and Pierre Lavency. Preferences; putting more knowledge into queries. In *VLDB '87: Proceedings of the 13th International Conference on Very Large Data Bases*, pages 217–225, San Francisco, CA, USA, 1987. Morgan Kaufmann Publishers Inc.

Mounia Lalmas, Andy MacFarlane, Stefan M. Rüger, Anastasios Tombros, Theodora Tsikrika, and Alexei Yavlinsky, editors. *Advances in Information Retrieval, 28th European Conference on IR Research, ECIR 2006, London, UK, April 10-12, 2006, Proceedings*, volume 3936 of *Lecture Notes in Computer Science*, 2006. Springer. ISBN 3-540-33347-9.

Iosif Lazaridis, Qi Han, Xingbo Yu, Sharad Mehrotra, Nalini Venkatasubramanian, Dmitri V. Kalashnikov, and Weiwen Yang. Quasar: quality aware sensing architecture. *ACM SIGMOD Record*, 33(1):26–5, 2004. ISSN 0163-5808.

Ulf Leonhardt and Jeff Magee. Security considerations for a distributed location service. *Journal of Network and Systems Management*, 6(1):51–70, March 1998.

J. C. Lester and B. W. Porter. Developing and Empirically Evaluating Robust Explanation Generators: The KNIGHT Experiments. *Computational Linguistics*, 23(1):65–101, 1997.

Alon Y. Levy. Logic-based techniques in data integration. In *Logic-based artificial intelligence*, pages 575–595. Kluwer Academic Publishers, 2000. ISBN 0-7923-7224-7.

Henry A. Lieberman and T. Selker. Out of context: computer systems that adapt to, and learn from, context. *IBM Systems Journal*, 39(3-4):617–632, 2000.

Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tinydb: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1):122–173, 2005.

John McCarthy and Sasa Buvac. *Computing Natural Language : Formalizing Context (Expanded Notes)*, pages 13–37. Center for the Study of Language and Information, Stanford University, 1997.

M. F. McTear. User modelling for adaptive computer systems: A survey of recent developments. *AI Review*, 7:157–184, 1993.

Brian Meyers and Amanda Kern. <context-aware> schema </context-aware>. In *CHI Workshop on The What, Who, When, Where, Why, and How of Context-Awareness*, 2000.

M. Mutsuzaki, M. Theobald, Ander de Keijzer, Jennifer Widom, P. Agrawal, Omar Benjelloun, Anish Das Sarma, R. Murthy, and T. Sugihara. Trio-one: Layering uncertainty and lineage on a conventional dbms. In *Proc. of Conference on Innovative Data Systems Research, Monterey, USA*, pages 269–274, Online publication, January 2007. www.crdrrdb.org. ISBN not assigned.

William M. Newman, Marge Eldridge, and Mik Lamming. Pepys: Generating autobiographies by automatic tracking. In *Second European Conf. on Computer-Supported Cooperative Work*, pages 175–188, 1991.

OWL Part-whole(2008). Simple part-whole relations in owl ontologies. <http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/>, August 2005. Editor’s Draft, work in progress.

A. Paramythis and Stephan Weibelzahl. A Decomposition Model for the Layered Evaluation of Interactive Adaptive Systems. *User Modeling 2005: 10th International Conference, UM 2005, Edinburgh, Scotland, UK, July 24-29, 2005: Proceedings*, 2005.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988. ISBN 1558604790.

Martin Peim, Enrico Franconi, Norman W. Paton, and Carole A. Goble. Query processing with description logic ontologies over object-wrapped databases. In *the 14th International Conference on Scientific and Statistical Database Management (SSDBM-2002)*, pages 27–36. IEEE Computer Society, July 2002.

Pellet. Pellet. <http://pellet.owldl.com/>, 2008.

C. Raymond Perrault, James F. Allen, and Philip R. Cohen. Speech acts as a basis for understanding dialogue coherence. In *Proceedings of the 1978 workshop on Theoretical issues in natural language processing*, pages 125–132, Morristown, NJ, USA, 1978. Association for Computational Linguistics.

Jay M. Ponte and Bruce W. Croft. A language modeling approach to information retrieval. In *SIGIR*, pages 275–281. ACM, 1998.

R. W. Poppe and R. J. Rienks. Evaluating the future of hci: challenges for the evaluation of upcoming applications. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'07) workshop on Artificial Intelligence for Human Computing (AI4HC)*, pages 89–96, Hyderabad, India, 2007. IJCAI. ISBN=not assigned.

Davy Preuveneers, Jan Van Bergh den, Dennis Wagelaar, Andy Georges, Peter Rigole, Tim Clerckx, Yolande Berbers, Karin Coninx, Viviane Jonckers, and Koen De Bosschere. Towards an extensible context ontology for ambient intelligence. In *Ambient Intelligence: Second European Symposium, EUSAI 2004, Eindhoven, The Netherlands, November 8-11, 2004. Proceedings*, pages 148–159. Springer, 2004.

Protégé (2008). Protégé. <http://protege.stanford.edu/>, 2008.

Racer (2008). Racer. <http://www.racer-systems.com/>, 2008.

Georgina Ramírez and Arjen P. de Vries. Relevant contextual features in xml retrieval. In *IIX: Proceedings of the 1st international conference on Information interaction in context*, pages 56–65, New York, NY, USA, 2006. ACM Press.

Anand Ranganathan, Jalal Al-Muhtadi, and Roy H. Campbell. Reasoning about uncertain contexts in pervasive computing environments. *IEEE Pervasive Computing*, 3(2):62–70, 2004.

Christopher Ré, Nilesh N. Dalvi, and Dan Suciu. Query evaluation on probabilistic databases. *IEEE Data Eng. Bull.*, 29(1):25–31, 2006.

Alan Rector. Medical informatics. In Franz Baader, Diego Calvanese, D.L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, *The Description Logic Handbook*, pages 406–426. Cambridge University Press, November 2003. ISBN 0-521-78176-0.

Elaine Rich. Users are individuals: Individualizing user models. *International Journal of Man-Machine Studies*, 18(3):199–214, 1983.

Elaine Rich. User modeling via stereotypes. *Readings in intelligent user interfaces*, pages 329–342, 1998.

Henning Rode and Djoerd Hiemstra. Using query profiles for clarification. In Lalmas et al. (2006), pages 205–216. ISBN 3-540-33347-9.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition edition, 2004. ISBN 0130803022.

Anish Das Sarma, M. Theobald, and Jennifer Widom. Exploiting lineage for confidence computation in uncertain and probabilistic databases. Technical report, Stanford University, March 2007.

Mahadev Satyanarayanan. Accessing information on demand at any location: Mobile information access. *IEEE Personal Communications*, 3(1): 26–33, February 1996. ISSN 1070-9916.

Mahadev Satyanarayanan. Pervasive computing: Vision and challenges. *IEEE Personal Communications*, 8:10–7, 2001.

Bill N. Schilit. *A Context-Aware System Architecture for Mobile Distributed Computing*. PhD thesis, Columbia University, May 1995.

Bill N. Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *the Workshop on Mobile Computing Systems and Applications*, pages 85–90. IEEE Computer Society, 1994.

Albrecht Schmidt. There is more to context than location. *Computers and Graphics Journal*, 23(6):893–901, 1999.

Glenn Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

Xuehua Shen, Bin Tan, and ChengXiang Zhai. Context-sensitive information retrieval using implicit feedback. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 43–50, New York, NY, USA, 2005. ACM Press.

Alkis Simitsis and Georgia Koutrika. Comprehensible answers to précis queries. In Eric Dubois and Klaus Pohl, editors, *CAiSE*, volume 4001 of *Lecture Notes in Computer Science*, pages 142–156. Springer, 2006. ISBN 3-540-34652-X.

Alkis Simitsis, Georgia Koutrika, and Yannis E. Ioannidis. Précis: from unstructured keywords as queries to structured databases as answers. *VLDB J.*, accepted for pub, 2008.

Robert F. Simmons. Natural language question-answering systems: 1969. *Commun. ACM*, 13(1):15–30, 1970. ISSN 0001-0782.

Kostas Stefanidis and Evaggelia Pitoura. Fast contextual preference scoring of database tuples. In *EDBT 2008*, 2008.

Kostas Stefanidis, Evaggelia Pitoura, and Panos Vassiliadis. On supporting context-aware preferences in relational database systems. In *First International Workshop on Managing Context Information in Mobile and Pervasive Environments (MCMP'2005)*, May 2005.

Kostas Stefanidis, Evaggelia Pitoura, and Panos Vassiliadis. Adding context to preferences. In *ICDE 2007*, April 2007.

Thomas Strang. A context modeling survey. In *Workshop on Advanced Context Modelling, Reasoning and Management associated with the Sixth International Conference on Ubiquitous Computing (UbiComp 2004)*, September 2004.

Dan Suciu and Nilesch N. Dalvi. Foundations of probabilistic answers to queries. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, page 963, New York, NY, USA, 2005. ACM Press.

Jean Tague-Sutcliffe. The pragmatics of information retrieval experimentation, revisited. *Inf. Process. Manage.*, 28(4):467–490, 1992. ISSN 0306-4573.

David Tennenhouse. Proactive computing. *Communications of the ACM*, 43(5):43–50, 2000. ISSN 0001-0782.

Nava Tintarev and Judith Masthoff. A survey of explanations in recommender systems. In *Workshop on Recommender Systems and Intelligent User Interfaces associated with ICDE'07*, 2007.

Arthur Hugo van Bunningen and Maarten Fokkinga. Possible histories: A way to model context-aware preferences. Technical Report TR-CTIT-06-78, University of Twente, Enschede, December 2006.

Harold van Heerde, Nicolas Anciaux, Maarten Fokkinga, and Peter M. G. Apers. Exploring personalized life cycle policies. Technical Report TR-CTIT-07-85, University of Twente, Enschede, December 2007.

Xiao Hang Wang, Tao Gu, Da Qing Zhang, and Hung Keng Pung. Ontology based context modeling and reasoning using owl. In *2nd IEEE Conference on Pervasive Computing and Communications, Workshop on Context Modeling and Reasoning*, pages 18–22. IEEE Computer Society, 2004.

Stephan Weibelzahl and Gerhard Weber. Advantages, opportunities and limits of empirical evaluations: Evaluating adaptive systems. *KI*, 16(3): 17–20, 2002.

Mark Weiser. The computer for the 21st century. *Scientific American*, 265(3):94–104, 1991.

Timo Weithoner, Thorsten Liebig, and Gunther Specht. Storing and querying ontologies in logic databases. In *Proceedings of The first International Workshop on Semantic Web and Databases (SWDB'03)*, pages 329–348, 2003.

Ji-Rong Wen, Ni Lao, and Wei-Ying Ma. Probabilistic model for contextual retrieval. In *Proceedings of the 27th annual international conference on Research and development in information retrieval*, pages 57–63. ACM Press, Sheffield, United Kingdom, 2004. ISBN 1-58113-881-4.

Kamin Whitehouse, Feng Zhao, and Jie Liu. Semantic streams: a framework for declarative queries and automatic data interpretation. Technical Report MSR-TR-2005-45, Microsoft Research, One Microsoft Way, Redmond, WA 98052, April 2005.

Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. *Proceedings of the Fourteenth International Conference on Machine Learning*, 97, 1997.

Samenvatting

In een razend tempo zijn we in staat om nieuwe gegevens te digitaliseren. Een mogelijke techniek om te voorkomen dat gebruikers de weg kwijt raken in deze hoeveelheid informatie is contextbewustheid; de beschikbaar gestelde informatie, of het gedrag van een systeem, aanpassen aan de huidige situatie van de gebruiker. Onderzocht is hoe contextbewuste applicaties zijn te ondersteunen met behulp van een applicatie-onafhankelijk raamwerk. De kern van dit raamwerk is het aan de context van de gebruiker aanpassen van de antwoorden op vragen die een gebruiker aan een gegevensbank stelt.

Door de manier waarop de context van een gebruiker vaak wordt gemeten (met behulp van sensoren) en gebruikt (om proactief informatie aan de gebruiker te doen toekomen), zijn er verscheidene onderwerpen die specifiek voor dit soort data aan de orde gesteld moeten worden. Vier onderwerpen worden expliciet onderzocht, te weten, de mogelijkheid om een antwoord terug te voeren tot het waarom, het omgaan met onzekerheid in de contextdata, de mogelijkheid om het gedrag van de gebruiker te leren en de mogelijkheid om de contextdata op te slaan.

Rekening houdend met deze onderwerpen is een model, genaamd CIM, ontwikkeld waarin kan worden vastgelegd hoe contextdata de informatiebehoefte van een gebruiker beïnvloedt en op basis waarvan de antwoorden van een gegevensbank kunnen worden geordend op relevantie, of waarmee proactief de meest relevante antwoorden aan de gebruiker kunnen worden aangeboden. Dit model slaat de effecten van context op de gebruiker op doormiddel van voorkeursregels voorzien van een score, tezamen met een score-combinatie-operator. Iedere regel geeft een interesse van een gebruiker aan in een specifieke context. Zowel de interesses, als de context worden gerepresenteerd in *Description Logics*, om zo in staat te zijn redeneermechanismes te ondersteunen en de regels voor gebruikers begrijpbaar te houden. De combinatie-operator bepaalt wat de (sterkte van) de voorkeur van de ge-

bruiker is als een document voldoet aan meerdere interesses. Er worden twee combinatie-operators voorgesteld: een welke een conjunctieve relatie tussen voorkeuren veronderstelt en een tweede welke een disjunctieve relatie veronderstelt. Beide gaan uit van een (verschillende) probabilistische interpretatie van de scores.

De begripbaarheid van de scores in het model voor de gebruikers blijkt uit het feit dat indien gebruikers zelf de scores van hun voorkeursregels kunnen aanpassen, dit inderdaad leidt tot een beter antwoord van het systeem dan wanneer slechts een ordering van voorkeursregels door de gebruikers is gegeven. Verder is onderzocht welke (onzekere) aannames het onderliggende systeem als uitleg van een antwoord zou moeten presenteren om de gebruiker de kans te geven, door verificatie van deze aannames, meer zekerheid te krijgen over het antwoord. Hiervoor worden twee maten voor de kwaliteit van een uitleg voor een antwoord geïntroduceerd.

Het model kan eenvoudig omgaan met opgeslagen contextdata via een vertaling van het model naar relationele databases. Het omgaan met onzekerheid in de contextdata en het effect hiervan op een goede ordering, wordt gedaan via *probabilistic event expressions*. Ook wordt laten zien hoe scores van voorkeursregels te leren zijn uit de interactie-geschiedenis van een gebruiker met het systeem. Testen met gesimuleerde gebruikers tonen aan dat indien conjunctief of disjunctief gedrag in de geschiedenis aanwezig is, deze scores ook te leren zijn, maar ook scores van echte gebruikers zijn deels te leren op zo'n manier dat een representatie met geleerde scores beter geordende antwoorden geeft dan een willekeurige representatie.

Een belangrijk inzicht dat ons model geeft is hoe de aanpak van onzekerheid, scores van voorkeuren voor het disjunctieve gedrag en inzicht geven in antwoorden, hand in hand gaan door middel van probabilistische event expressies. Er zijn echter twee belangrijke punten die meegenomen moeten worden alvorens overtuigend te kunnen zeggen dat de beschreven methode in deze thesis daadwerkelijk toepasbaar is in systemen op grote schaal. Deze hangen samen met het exploratieve karakter van het onderzoek en bestaan uit het daadwerkelijk testen van de technieken op grote schaal en het daarbij ook rekening houden met de niet expliciet aan de orde gestelde onderwerpen in deze thesis (zoals distributiviteit).

SIKS Dissertatiereeks

- 1998-1 Johan van den Akker (CWI)
DEGAS - An Active, Temporal Database of Autonomous Objects
- 1998-2 Floris Wiesman (UM)
Information Retrieval by Graphically Browsing Meta-Information
- 1998-3 Ans Steuten (TUD)
A Contribution to the Linguistic Analysis of Business Conversations
within the Language/Action Perspective
- 1998-4 Dennis Breuker (UM)
Memory versus Search in Games
- 1998-5 E.W.Oskamp (RUL)
Computerondersteuning bij Straftoemeting
-
- 1999-1 Mark Sloof (VU)
Physiology of Quality Change Modelling; Automated modelling
of Quality Change of Agricultural Products
- 1999-2 Rob Potharst (EUR)
Classification using decision trees and neural nets
- 1999-3 Don Beal (UM)
The Nature of Minimax Search
- 1999-4 Jacques Penders (UM)
The practical Art of Moving Physical Objects
- 1999-5 Aldo de Moor (KUB)
Empowering Communities: A Method for the Legitimate User-Driven
Specification of Network Information Systems
- 1999-6 Niek J.E. Wijngaards (VU)
Re-design of compositional systems
- 1999-7 David Spelt (UT)
Verification support for object database design
- 1999-8 Jacques H.J. Lenting (UM)
Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism
for Discrete Reallocation.
-
- 2000-1 Frank Niessink (VU)
Perspectives on Improving Software Maintenance
- 2000-2 Koen Holtman (TUE)
Prototyping of CMS Storage Management
- 2000-3 Carolien M.T. Metselaar (UVA)
Sociaal-organisatorische gevolgen van kennistechnologie;
een procesbenadering en actorperspectief.
- 2000-4 Geert de Haan (VU)
ETAG, A Formal Model of Competence Knowledge for User Interface
Design

- 2000-5 Ruud van der Pol (UM)
Knowledge-based Query Formulation in Information Retrieval.
- 2000-6 Rogier van Eijk (UU)
Programming Languages for Agent Communication
- 2000-7 Niels Peek (UU)
Decision-theoretic Planning of Clinical Patient Management
- 2000-8 Veerle Coupé (EUR)
Sensitivity Analysis of Decision-Theoretic Networks
- 2000-9 Florian Waas (CWI)
Principles of Probabilistic Query Optimization
- 2000-10 Niels Nes (CWI)
Image Database Management System Design Considerations, Algorithms and Architecture
- 2000-11 Jonas Karlsson (CWI)
Scalable Distributed Data Structures for Database Management

- 2001-1 Silja Renooij (UU)
Qualitative Approaches to Quantifying Probabilistic Networks
- 2001-2 Koen Hindriks (UU)
Agent Programming Languages: Programming with Mental Models
- 2001-3 Maarten van Someren (UvA)
Learning as problem solving
- 2001-4 Evgueni Smirnov (UM)
Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets
- 2001-5 Jacco van Ossenbruggen (VU)
Processing Structured Hypermedia: A Matter of Style
- 2001-6 Martijn van Welie (VU)
Task-based User Interface Design
- 2001-7 Bastiaan Schonhage (VU)
Diva: Architectural Perspectives on Information Visualization
- 2001-8 Pascal van Eck (VU)
A Compositional Semantic Structure for Multi-Agent Systems Dynamics.
- 2001-9 Pieter Jan 't Hoen (RUL)
Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes
- 2001-10 Maarten Sierhuis (UvA)
Modeling and Simulating Work Practice
BRAHMS: a multiagent modeling and simulation language for work practice analysis and design
- 2001-11 Tom M. van Engers (VUA)
Knowledge Management:
The Role of Mental Models in Business Systems Design

- 2002-01 Nico Lassing (VU)
Architecture-Level Modifiability Analysis
- 2002-02 Roelof van Zwol (UT)
Modelling and searching web-based document collections
- 2002-03 Henk Ernst Blok (UT)
Database Optimization Aspects for Information Retrieval
- 2002-04 Juan Roberto Castelo Valdueza (UU)
The Discrete Acyclic Digraph Markov Model in Data Mining
- 2002-05 Radu Serban (VU)
The Private Cyberspace Modeling Electronic Environments inhabited by Privacy-concerned Agents
- 2002-06 Laurens Mommers (UL)
Applied legal epistemology; Building a knowledge-based ontology of the legal domain
- 2002-07 Peter Boncz (CWI)
Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications
- 2002-08 Jaap Gordijn (VU)

- Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas
- 2002-09 Willem-Jan van den Heuvel (KUB)
Integrating Modern Business Applications with Objectified Legacy Systems
- 2002-10 Brian Sheppard (UM)
Towards Perfect Play of Scrabble
- 2002-11 Wouter C.A. Wijngaards (VU)
Agent Based Modelling of Dynamics: Biological and Organisational Applications
- 2002-12 Albrecht Schmidt (Uva)
Processing XML in Database Systems
- 2002-13 Hongjing Wu (TUE)
A Reference Architecture for Adaptive Hypermedia Applications
- 2002-14 Wieke de Vries (UU)
Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems
- 2002-15 Rik Eshuis (UT)
Semantics and Verification of UML Activity Diagrams for Workflow Modelling
- 2002-16 Pieter van Langen (VU)
The Anatomy of Design: Foundations, Models and Applications
- 2002-17 Stefan Manegold (UVA)
Understanding, Modeling, and Improving Main-Memory Database Performance
- 2003-01 Heiner Stuckenschmidt (VU)
Ontology-Based Information Sharing in Weakly Structured Environments
- 2003-02 Jan Broersen (VU)
Modal Action Logics for Reasoning About Reactive Systems
- 2003-03 Martijn Schuemie (TUD)
Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy
- 2003-04 Milan Petkovic (UT)
Content-Based Video Retrieval Supported by Database Technology
- 2003-05 Jos Lehmann (UVA)
Causation in Artificial Intelligence and Law - A modelling approach
- 2003-06 Boris van Schooten (UT)
Development and specification of virtual environments
- 2003-07 Machiel Jansen (UvA)
Formal Explorations of Knowledge Intensive Tasks
- 2003-08 Yongping Ran (UM)
Repair Based Scheduling
- 2003-09 Rens Kortmann (UM)
The resolution of visually guided behaviour
- 2003-10 Andreas Lincke (UvT)
Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture
- 2003-11 Simon Keizer (UT)
Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks
- 2003-12 Roeland Ordelman (UT)
Dutch speech recognition in multimedia information retrieval
- 2003-13 Jeroen Donkers (UM)
Nosce Hostem - Searching with Opponent Models
- 2003-14 Stijn Hoppenbrouwers (KUN)
Freezing Language: Conceptualisation Processes across ICT-Supported Organisations
- 2003-15 Mathijs de Weerd (TUD)
Plan Merging in Multi-Agent Systems
- 2003-16 Menzo Windhouwer (CWI)
Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouses
- 2003-17 David Jansen (UT)
Extensions of Statecharts with Probability, Time, and Stochastic Timing
- 2003-18 Levente Kocsis (UM)
Learning Search Decisions
- 2004-01 Virginia Dignum (UU)
A Model for Organizational Interaction: Based on Agents, Founded in Logic

- 2004-02 Lai Xu (UvT)
Monitoring Multi-party Contracts for E-business
- 2004-03 Perry Groot (VU)
A Theoretical and Empirical Analysis of Approximation
in Symbolic Problem Solving
- 2004-04 Chris van Aart (UVA)
Organizational Principles for Multi-Agent Architectures
- 2004-05 Viara Popova (EUR)
Knowledge discovery and monotonicity
- 2004-06 Bart-Jan Hommes (TUD)
The Evaluation of Business Process Modeling Techniques
- 2004-07 Elise Boltjes (UM)
Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract
denken, vooral voor meisjes
- 2004-08 Joop Verbeek(UM)
Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politieke
gegevensuitwisseling en digitale expertise
- 2004-09 Martin Caminada (VU)
For the Sake of the Argument; explorations into argument-based reasoning
- 2004-10 Suzanne Kabel (UVA)
Knowledge-rich indexing of learning-objects
- 2004-11 Michel Klein (VU)
Change Management for Distributed Ontologies
- 2004-12 The Duy Bui (UT)
Creating emotions and facial expressions for embodied agents
- 2004-13 Wojciech Jamroga (UT)
Using Multiple Models of Reality: On Agents who Know how to Play
- 2004-14 Paul Harrenstein (UU)
Logic in Conflict. Logical Explorations in Strategic Equilibrium
- 2004-15 Arno Knobbe (UU)
Multi-Relational Data Mining
- 2004-16 Federico Divina (VU)
Hybrid Genetic Relational Search for Inductive Learning
- 2004-17 Mark Winands (UM)
Informed Search in Complex Games
- 2004-18 Vania Bessa Machado (UvA)
Supporting the Construction of Qualitative Knowledge Models
- 2004-19 Thijs Westerveld (UT)
Using generative probabilistic models for multimedia retrieval
- 2004-20 Madelon Evers (Nyenrode)
Learning from Design: facilitating multidisciplinary design teams
- 2005-01 Floor Verdenius (UVA)
Methodological Aspects of Designing Induction-Based Applications
- 2005-02 Erik van der Werf (UM)
AI techniques for the game of Go
- 2005-03 Franc Grootjen (RUN)
A Pragmatic Approach to the Conceptualisation of Language
- 2005-04 Nirvana Meratnia (UT)
Towards Database Support for Moving Object data
- 2005-05 Gabriel Infante-Lopez (UVA)
Two-Level Probabilistic Grammars for Natural Language Parsing
- 2005-06 Pieter Spronck (UM)
Adaptive Game AI
- 2005-07 Flavius Frasincar (TUE)
Hypermedia Presentation Generation for Semantic Web Information Systems
- 2005-08 Richard Vdovjak (TUE)
A Model-driven Approach for Building Distributed Ontology-based Web
Applications
- 2005-09 Jeen Broekstra (VU)
Storage, Querying and Inferencing for Semantic Web Languages
- 2005-10 Anders Bouwer (UVA)

- Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments
2005-11 Elth Ogston (VU)
Agent Based Matchmaking and Clustering - A Decentralized Approach to Search
- 2005-12 Csaba Boer (EUR)
Distributed Simulation in Industry
- 2005-13 Fred Hamburg (UL)
Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen
- 2005-14 Borys Omelayenko (VU)
Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics
- 2005-15 Tibor Bosse (VU)
Analysis of the Dynamics of Cognitive Processes
- 2005-16 Joris Graaumanns (UU)
Usability of XML Query Languages
- 2005-17 Boris Shishkov (TUD)
Software Specification Based on Re-usable Business Components
- 2005-18 Danielle Sent (UU)
Test-selection strategies for probabilistic networks
- 2005-19 Michel van Dartel (UM)
Situating Representation
- 2005-20 Cristina Coteanu (UL)
Cyber Consumer Law, State of the Art and Perspectives
- 2005-21 Wijnand Derks (UT)
Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics
- 2006-01 Samuil Angelov (TUE)
Foundations of B2B Electronic Contracting
- 2006-02 Cristina Chisalita (VU)
Contextual issues in the design and use of information technology in organizations
- 2006-03 Noor Christoph (UVA)
The role of metacognitive skills in learning to solve problems
- 2006-04 Marta Sabou (VU)
Building Web Service Ontologies
- 2006-05 Cees Pierik (UU)
Validation Techniques for Object-Oriented Proof Outlines
- 2006-06 Ziv Baida (VU)
Software-aided Service Bundling - Intelligent Methods & Tools for Graphical Service Modeling
- 2006-07 Marko Smiljanic (UT)
XML schema matching -- balancing efficiency and effectiveness by means of clustering
- 2006-08 Eelco Herder (UT)
Forward, Back and Home Again - Analyzing User Behavior on the Web
- 2006-09 Mohamed Wahdan (UM)
Automatic Formulation of the Auditor's Opinion
- 2006-10 Ronny Siebes (VU)
Semantic Routing in Peer-to-Peer Systems
- 2006-11 Joeri van Ruth (UT)
Flattening Queries over Nested Data Types
- 2006-12 Bert Bongers (VU)
Interactivation - Towards an e-cology of people, our technological environment, and the arts
- 2006-13 Henk-Jan Lebbink (UU)
Dialogue and Decision Games for Information Exchanging Agents
- 2006-14 Johan Hoorn (VU)
Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change
- 2006-15 Rainer Malik (UU)
CONAN: Text Mining in the Biomedical Domain
- 2006-16 Carsten Riggelsen (UU)
Approximation Methods for Efficient Learning of Bayesian Networks
- 2006-17 Stacey Nagata (UU)
User Assistance for Multitasking with Interruptions on a Mobile Device
- 2006-18 Valentin Zhizhikun (UVA)

- Graph transformation for Natural Language Processing
2006-19 Birna van Riemsdijk (UU)
Cognitive Agent Programming: A Semantic Approach
- 2006-20 Marina Velikova (UvT)
Monotone models for prediction in data mining
- 2006-21 Bas van Gils (RUN)
Aptness on the Web
- 2006-22 Paul de Vrieze (RUN)
Fundaments of Adaptive Personalisation
- 2006-23 Ion Juvina (UU)
Development of Cognitive Model for Navigating on the Web
- 2006-24 Laura Hollink (VU)
Semantic Annotation for Retrieval of Visual Resources
- 2006-25 Madalina Drugan (UU)
Conditional log-likelihood MDL and Evolutionary MCMC
- 2006-26 Vojkan Mihajlovic (UT)
Score Region Algebra: A Flexible Framework for Structured Information Retrieval
- 2006-27 Stefano Bocconi (CWI)
Vox Populi: generating video documentaries from semantically annotated media repositories
- 2006-28 Borkur Sigurbjornsson (UVA)
Focused Information Access using XML Element Retrieval
- 2007-01 Kees Leune (UvT)
Access Control and Service-Oriented Architectures
- 2007-02 Wouter Teepe (RUG)
Reconciling Information Exchange and Confidentiality: A Formal Approach
- 2007-03 Peter Mika (VU)
Social Networks and the Semantic Web
- 2007-04 Jurriaan van Diggelen (UU)
Achieving Semantic Interoperability in Multi-agent Systems: a dialogue-based approach
- 2007-05 Bart Schermer (UL)
Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance
- 2007-06 Gilad Mishne (UVA)
Applied Text Analytics for Blogs
- 2007-07 Natasa Jovanovic' (UT)
To Whom It May Concern - Addressee Identification in Face-to-Face Meetings
- 2007-08 Mark Hoogendoorn (VU)
Modeling of Change in Multi-Agent Organizations
- 2007-09 David Mobach (VU)
Agent-Based Mediated Service Negotiation
- 2007-10 Huib Aldewereld (UU)
Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols
- 2007-11 Natalia Stash (TUE)
Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System
- 2007-12 Marcel van Gerven (RUN)
Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty
- 2007-13 Rutger Rienks (UT)
Meetings in Smart Environments; Implications of Progressing Technology
- 2007-14 Niek Bergboer (UM)
Context-Based Image Analysis
- 2007-15 Joyca Lacroix (UM)
NIM: a Situated Computational Memory Model
- 2007-16 Davide Grossi (UU)
Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi-agent Systems
- 2007-17 Theodore Charitos (UU)
Reasoning with Dynamic Networks in Practice

- 2007-18 Bart Orriens (UvT)
On the development and management of adaptive business collaborations
- 2007-19 David Levy (UM)
Intimate relationships with artificial partners
- 2007-20 Slinger Jansen (UU)
Customer Configuration Updating in a Software Supply Network
- 2007-21 Karianne Vermaas (UU)
Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005
- 2007-22 Zlatko Zlatev (UT)
Goal-oriented design of value and process models from patterns
- 2007-23 Peter Barna (TUE)
Specification of Application Logic in Web Information Systems
- 2007-24 Georgina Ramírez Camps (CWI)
Structural Features in XML Retrieval
- 2007-25 Joost Schalken (VU)
Empirical Investigations in Software Process Improvement
- 2008-01 Katalin Boer-Sorbán (EUR)
Agent-Based Simulation of Financial Markets: A modular, continuous-time approach
- 2008-02 Alexei Sharpanskykh (VU)
On Computer-Aided Methods for Modeling and Analysis of Organizations
- 2008-03 Vera Hollink (UVA)
Optimizing hierarchical menus: a usage-based approach
- 2008-04 Ander de Keijzer (UT)
Management of Uncertain Data - towards unattended integration
- 2008-05 Bela Mutschler (UT)
Modeling and simulating causal dependencies on process-aware information systems from a cost perspective
- 2008-06 Arjen Hommersom (RUN)
On the Application of Formal Methods to Clinical Guidelines, an Artificial Intelligence Perspective
- 2008-07 Peter van Rosmalen (OU)
Supporting the tutor in the design and support of adaptive e-learning
- 2008-08 Janneke Bolt (UU)
Bayesian Networks: Aspects of Approximate Inference
- 2008-09 Christof van Nimwegen (UU)
The paradox of the guided user: assistance can be counter-effective
- 2008-10 Wouter Bosma (UT)
Discourse oriented summarization
- 2008-11 Vera Kartseva (VU)
Designing Controls for Network Organizations: A Value-Based Approach
- 2008-12 Jozsef Farkas (RUN)
A Semiotically Oriented Cognitive Model of Knowledge Representation
- 2008-13 Caterina Carraciolo (UVA)
Topic Driven Access to Scientific Handbooks
- 2008-14 Arthur van Bunningen (UT)
Context-Aware Querying; Better Answers with Less Effort
- 2008-15 Martijn van Otterlo (UT)
The Logic of Adaptive Behavior: Knowledge Representation and Algorithms for the Markov Decision Process Framework in First-Order Domains.