

**Fakultät für Informatik
der Technischen Universität München**

LINGUINI

—

**Acquiring Individual Interest Profiles
by Means of Adaptive Natural Language Dialog**

Rosmary Stegmann

Fakultät für Informatik
der Technischen Universität München

Lehrstuhl Informatik XI

LINGUINI

—

Acquiring Individual Interest Profiles
by Means of Adaptive Natural Language Dialog

Rosmary Stegmann

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangungen des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. Uwe Baumgarten

Prüfer der Dissertation: 1. Univ.-Prof. Dr. Johann Schlichter

2. Univ.-Prof. Dr. Manfred Pinkal
Universität des Saarlandes

Die Dissertation wurde am 30.03.2006 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 19.07.2006 angenommen.

Abstract

User information is needed by adaptive systems in order to tailor information and product offers to the needs and preferences of individual users. Personalized Recommender Systems are adaptive systems that automatically generate recommendations on the basis of individual user profiles. Most existing Recommender Systems, however, are based on rather simple and mainly standardized profile information, which often delimits the adequacy of the recommendations they generate for an individual user. More adequate recommendations could be generated on the basis of more individual and representative user profiles that also integrate complex information, for example about personal interests or lifestyle. Furthermore, most adaptive systems acquire profile information only for their own purposes and do not allow for an exchange of this information with other applications the user wants to use. Above all, existing explicit profiling methods suffer from severe drawbacks which limit their utilizability in practice. Especially for mobile scenarios, in which a spoken language interaction with the user is required, no suitable explicit profiling methods exist as yet that integrate a solution for all of the above mentioned problems.

This thesis presents a solution for acquiring detailed information about personal interests of users by means of an adaptive natural language dialog. We have developed a comprehensive explicit profiling framework, LINGUINI, which integrates a dialog management and profile management approach. Because of the natural language processing methods applied, this profiling approach is especially suitable for situations in which spoken language is required (e.g. in a vehicle), but it is also applicable with a user interface for typed input and output (e.g. for Internet and E-Commerce platforms). The acquired information can be used by various types of adaptive systems for which user interests are relevant.

During our profiling dialog, users are able to formulate their interests in their own words. The dialog adapts to each user individually and is able to find and talk about new interests related to the interests already mentioned by the user. The dialog management approach integrates a sociological target group model that clusters users into groups according to their interests. The groups do not serve as user profiles, however, but are used for providing clues about suitable next questions or related topics. With this adaptive approach, we are able to create truly personalized profiles that are different for each user in contents and structure. By employing the lexical-semantic network GermaNet, our profiling approach allows for representing interests in a semantically structured way and for interpreting and storing new user information dynamically that has not been predefined in the user model before.

We implemented our adaptive profiling approach as a comprehensive prototype system and evaluated it by means of a user study which investigates user acceptance, dialog adaptability, and profile quality. The study shows that users, in fact, appreciate the adaptive capabilities of the profiling system. The users' willingness to apply the system is high and they consider this approach very suitable for a variety of mobile and non-mobile situations and adaptive applications.

Acknowledgements

This work emerged from my activities as a research assistant at the chair for Informatics XI: Applied Informatics/Cooperative Systems (Prof. Johann Schlichter) at the Technische Universität München.

First, I would like to thank my supervisor Johann Schlichter for providing the opportunity to work on this interesting research topic, and my second supervisor Manfred Pinkal, especially for his support with respect to the Natural Language Processing aspects of this work. Their helpful suggestions and comments greatly contributed to the success of this thesis.

I also thank my colleagues for the friendly and cooperative working atmosphere. In particular I would like to thank Wolfgang Wörndl for constantly encouraging me and for discussing and commenting on large parts of this work. I also thank Michael Koch, from whom I learned a lot, for helping me with the initiation of this project. I cannot thank enough Georg Groh and Michael Galla for their support during the preparation of my mathematics exam, which was a prerequisite for the dissertation in Informatics. Special thanks also to Thomas Leckner for the excellent cooperation during several years of common work on personalization and recommender systems.

I am very grateful to the students who contributed to the prototype implementation of LINGUINI, especially to the incomparable commitment of Kristof Unterweger and Manuel Giuliani. Many thanks also to the participants of the user study for patiently answering LINGUINI's nosy questions and for the multitude of valuable comments and suggestions they provided in their questionnaires.

I would also like to thank the researchers from other institutions for their support and the many inspiring and visionary discussions, which encouraged me to dare the impossible. Representative for all of them, these are Mathias Bauer, Aljoscha Burchardt, Katrin Erk, Alexander Felfernig, Gerd Fliedner, Iryna Gurevych, Anne Hackett, Dominik Heckmann, Elco Herder, Vera Hollink, Anthony Jameson, Ivana Kruijff-Korbayova, Jonas Kuhn, Claudia Kunze, Martin Lacher, Lothar Lemnitzer, Alexandros Paramythis, Bodo Polzer, Helmut Schmid, Sebastian Stegmann, Stefan Weibelzahl, and Ingrid Zukerman.

I am very grateful to Colleen Gruban who helped me proofread this work and improve my English. Last but not least, my deep gratitude belongs to Patrick, my family and friends for their understanding and great support in all situations.

Contents

LIST OF FIGURES	XI
LIST OF TABLES	XIII
LIST OF ACRONYMS	XV
1 INTRODUCTION.....	1
1.1 BACKGROUND	1
1.1.1 Adaptive Systems and Personalized Recommender Systems	1
1.1.2 User Models.....	1
1.1.3 Acquiring Information about Users.....	2
1.1.4 Acquiring Information and Generating Recommendations in a Mobile Application Scenario	3
1.2 MOTIVATION AND PROBLEM DESCRIPTION	4
1.3 GOALS OF THIS THESIS	6
1.4 METHODOLOGY AND STRUCTURE OF THE THESIS	7
1.4.1 Methodology.....	7
1.4.2 Structure	8
2 MODELING AND ACQUISITION OF USER PROFILES	9
2.1 USER MODELING AND APPLICATIONS	9
2.1.1 A Survey of User Modeling Approaches.....	9
2.1.1.1 User Models and Adaptive Systems	9
2.1.1.2 Technological Areas in which User Models Are Developed and Applied.....	10
2.1.1.3 Issues in User Modeling	13
2.1.1.4 Types of User Models.....	15
2.1.2 The User Modeling Approach of this Thesis	17
2.1.2.1 Modeling Individual Users and Using a Target Group Model for Dialog Control	18
2.1.2.2 Explicit Profile Information.....	19
2.1.2.3 Long-Term User Profiles for Different Applications.....	20
2.1.2.4 Personal Interests.....	20
2.1.2.5 Generating Profiles Dynamically on the Basis of a Lexical-Semantic Network.....	21
2.1.3 Applications Using Profiles: Personalized Recommender Systems.....	23
2.1.3.1 Introduction to Personalized Recommender Systems.....	23
2.1.3.2 Recommending Tourist Information.....	25
2.2 PROFILE ACQUISITION METHODS	27
2.2.1 Challenges for Acquiring Profile Information.....	27
2.2.2 Implicit and Explicit Profile Acquisition	28
2.2.2.1 Methods for Implicit Profiling.....	28
2.2.2.2 Methods for Explicit Profiling.....	30
2.2.3 Profiling Approaches and Adaptive Systems Involving Natural Language.....	34
2.2.3.1 User Profiling Chat Bots.....	34
2.2.3.2 Improving Natural Language Dialog Interaction by means of User Models	36
2.2.3.3 Using Dialog Systems for Collecting User Information and Offering Adaptive Services	38
2.2.4 The Profile Acquisition Method Developed for this Thesis	42
2.3 SUMMARY	44
3 NATURAL LANGUAGE PROCESSING METHODS AND RESOURCES APPLIED IN THIS THESIS	47
3.1 APPLYING NATURAL LANGUAGE PROCESSING FOR USER PROFILING	47
3.2 RECOGNITION AND SYNTHESIS OF SPOKEN LANGUAGE	49

3.2.1	<i>Speech Recognition (Speech-to-Text)</i>	49
3.2.2	<i>Speech Synthesis (Text-to-Speech)</i>	51
3.2.3	<i>Speech Recognition and Synthesis in LINGUINI</i>	52
3.3	PART-OF-SPEECH TAGGING	53
3.3.1	<i>Taggers and Tagsets</i>	53
3.3.2	<i>Choosing a Tagger for LINGUINI</i>	55
3.4	SYNTACTIC ANALYSIS.....	56
3.4.1	<i>Syntactic Parsing and Grammars</i>	56
3.4.2	<i>Choosing a Parser for LINGUINI</i>	58
3.5	LEXICAL-SEMANTIC NETWORKS	58
3.5.1	<i>GermaNet: A Lexical-Semantic Network for German</i>	59
3.5.2	<i>Using GermaNet for Profile Construction and Update in LINGUINI</i>	60
3.6	WORD SENSE DISAMBIGUATION	61
3.6.1	<i>The Problem of Ambiguity</i>	61
3.6.2	<i>Existing Approaches to Word Sense Disambiguation</i>	61
3.6.3	<i>Word Sense Disambiguation in LINGUINI</i>	63
3.7	SPEECH DIALOG SYSTEMS	66
3.7.1	<i>Dialog Management</i>	66
3.7.2	<i>In-Vehicle Application of Dialog Systems</i>	70
3.7.3	<i>Choosing a Dialog Strategy for LINGUINI</i>	72
3.8	NATURAL LANGUAGE GENERATION.....	73
3.8.1	<i>Natural Language Generation Approaches</i>	73
3.8.2	<i>The Natural Language Generation Approach of LINGUINI</i>	75
3.9	SUMMARY	77
4	OVERVIEW OF THE PROFILE ACQUISITION APPROACH.....	79
4.1	REQUIREMENTS FOR THE USER PROFILING SYSTEM	79
4.2	THE PROFILE ACQUISITION PROCESS	80
4.2.1	<i>The Process from a User's Point of View</i>	80
4.2.1.1	Constructing an Initial Profile.....	81
4.2.1.2	Refining the Profile during a Dialog Interaction.....	81
4.2.2	<i>The Process from a System-Internal Point of View</i>	84
4.2.2.1	Constructing an Initial Profile.....	84
4.2.2.2	Refining the Profile during a Dialog Interaction.....	85
4.3	THE SYSTEM ARCHITECTURE	90
4.4	THE PROFILE MODEL.....	91
4.4.1	<i>Interests and Dislikes Contained in Profile Instances</i>	92
4.4.2	<i>The Profile Graph and the Structure of Interest Nodes</i>	93
4.4.2.1	The Internal Structure of an Interest Node.....	94
4.4.2.2	Converting the GermaNet Graph into an XML Tree	96
4.4.2.3	Using GermaNet Nouns for the Profile Graph.....	98
4.5	SUMMARY	99
5	THE MAIN COMPONENTS OF THE PROFILE ACQUISITION SYSTEM.....	101
5.1	THE LANGUAGE ANALYSIS	101
5.1.1	<i>Overview of the Language Analysis</i>	101
5.1.1.1	Tasks and Embedding of the Language Analysis	101
5.1.1.2	The Semantic Classes Lexicon	103
5.1.2	<i>Functioning of the Language Analysis</i>	106
5.1.2.1	Language Analysis Classes.....	106
5.1.2.2	The Linguistic Analysis of User Answers	107
5.1.2.3	The Extraction of Attribute Values from the User Answer.....	108
5.1.2.4	The Creation of Interest Nodes.....	112
5.2	THE PROFILE MANAGER.....	114
5.2.1	<i>Overview of the Profile Manager: Tasks and Embedding</i>	114
5.2.2	<i>Functioning of the Profile Manager</i>	115
5.2.2.1	Profile Manager Classes	116
5.2.2.2	Constructing a Profile Graph	117
5.2.2.3	Augmenting and Updating the Profile	119
5.3	TARGET GROUP MATCHING AND TARGET GROUP MODEL	121
5.3.1	<i>Overview of the Target Group Matching</i>	121
5.3.1.1	Tasks and Embedding of the Target Group Matching	121
5.3.1.2	The Target Group Model	123

5.3.2	<i>Functioning of the Target Group Matching</i>	126
5.3.2.1	Target Group Matching Classes and Main Data Structures	126
5.3.2.2	The Main Method of the Target Group Matching.....	127
5.3.2.3	Extracting and Transforming Information from Profile Nodes and Target Group Nodes.....	128
5.3.2.4	Determining the Most Similar Target Group	129
5.4	THE DIALOG MANAGER	134
5.4.1	<i>Overview of the Dialog Manager</i>	134
5.4.1.1	Tasks and Embedding of the Dialog Manager	134
5.4.1.2	The Interaction between Dialog Manager and User Interface.....	136
5.4.2	<i>Functioning of the Dialog Manager</i>	136
5.4.2.1	Dialog Manager Classes and Main Data Structures.....	136
5.4.2.2	The Main Method of the Dialog Manager	139
5.4.2.3	Instructions for Answer Processing	142
5.4.2.4	Question Selection and Creation of Question Specifications.....	146
5.5	SUMMARY	156
6	IMPLEMENTATION AND EVALUATION	159
6.1	EMPIRICAL USER STUDY ABOUT PREFERRED USER INPUT	159
6.1.1	<i>Type and Aims of the User Study</i>	159
6.1.2	<i>Results</i>	160
6.1.3	<i>Discussion</i>	161
6.2	IMPLEMENTATION OF THE LINGUINI PROTOTYPE SYSTEM.....	161
6.2.1	<i>The Project Framework</i>	162
6.2.2	<i>Overview of the Prototype System</i>	162
6.2.2.1	Technologies Used for Implementing the Prototype System	162
6.2.2.2	Architecture and Functionality of the Prototype System	163
6.2.2.3	System Component Tests	163
6.2.3	<i>Components</i>	164
6.2.3.1	The Language Analysis	164
6.2.3.2	The Profile Manager.....	165
6.2.3.3	Target Group Matching and Target Group Model.....	166
6.2.3.4	The Dialog Manager.....	167
6.2.3.5	The Question Generation.....	168
6.2.3.6	The User Interface	168
6.3	FINAL EVALUATION OF PROFILING DIALOG AND PROFILE QUALITY	168
6.3.1	<i>A User Study about Dialog Adaptability, User Acceptance, and Development Status</i>	169
6.3.1.1	Type and Aims of the User Study.....	169
6.3.1.2	The Questionnaire.....	169
6.3.1.3	Preparation and Test Procedure	170
6.3.1.4	The Participants of the User Study	173
6.3.1.5	Results	173
6.3.2	<i>Profile Quality and Coverage Assessment</i>	183
6.3.2.1	Aims of the Quality and Coverage Assessment.....	183
6.3.2.2	Results	184
6.3.3	<i>Summary of Results and Discussion</i>	187
6.4	SUMMARY	191
7	CONCLUSIONS AND PROSPECTS	193
7.1	SUMMARY OF CONTRIBUTIONS	193
7.2	FUTURE WORK	195
7.2.1	<i>User Interfaces</i>	195
7.2.2	<i>The Visualization of User Profiles</i>	196
7.2.3	<i>Extension of the Core Functionality of the LINGUINI Prototype</i>	197
7.2.4	<i>Exchange of User Information and Privacy</i>	198
7.2.5	<i>Applications and Further Evaluations</i>	199
APPENDIX A: A SAMPLE TARGET GROUP: THE EXTENDED NIVEAUMILIEU	201	
APPENDIX B: THE FINAL SYSTEM EVALUATION	205	
B.1	THE QUESTIONNAIRE.....	205
B.2	USER GUIDELINES AND INSTRUCTIONS.....	210
B.3	SAMPLE DIALOGS	213
B.4	THE TREATMENT OF OUTLIER VALUES.....	218
BIBLIOGRAPHY	219	

List of Figures

2.1	Relations between user model, profile model, and profile instances	10
2.2	Target group model by Schulze [1992], extended with enjoyment attitudes . . .	15
2.3	Central storage and multiple usage of user profiles	20
2.4	Simplified sample interest node with attribute-value pairs	22
2.5	Specification of constraints and tour presentation in INTRIGUE	26
2.6	Syskill & Webert interface for rating pages	29
2.7	Lifestyle Finder – alias Waldo the Web Wizard – asking the user a question . . .	32
2.8	Cutout of a conversation with ALICE	35
2.9	Two interaction possibilities between dialog systems and user models	37
2.10	Dialog between COGITO chat bot Susanna and new customer with succeeding book recommendation	41
3.1	Natural language processing tasks in the LINGUINI profiling system	49
3.2	Possible parts-of-speech in order of frequency	53
3.3	Cutout of the Penn Treebank tagset	54
3.4	Sample output produced by the TreeTagger	56
3.5	Structural ambiguity	57
3.6	User input, tagged and chunked with TreeTagger and its associated chunker . . .	58
3.7	Cross-classification	59
3.8	Sample dialog with the Nuance automatic banking system	67
3.9	Sample dialog with the Philips Train Timetable Information System	68
3.10	Cutout of a dialog with Circuit-Fix-It shop	69
3.11	Simplified Question Specification with question frame and variable settings . . .	75
4.1	Cutout of a sample dialog with LINGUINI	83
4.2	System-internal steps during a single dialog unit	86
4.3	The LINGUINI system architecture	91
4.4	Cutout of a user profile instance	94
4.5	Cutout of the GermaNet graph for “Fußball (soccer)”	96
4.6	Cutout of a user profile represented in XML	97
4.7	Different GermaNet paths for “wandern” realized as noun or verb	98
5.1	Interaction between Language Analysis and Dialog Manager	102
5.2	Locations represented in a GermaNet XML file	104
5.3	Interaction between LA classes and external and internal resources	106
5.4	Language Analysis output examples	113

5.5	Interaction between Profile Manager and Dialog Manager	115
5.6	Interaction between PM classes, GermaNet, TreeTagger, and Profile Instances Database	116
5.7	Search algorithm for locating interest nodes in GermaNet	117
5.8	Profile node “Bücher lesen (reading books)”	120
5.9	A new attribute-value pair has been added	120
5.10	A new attribute value has been added	120
5.11	Interaction between Target Group Matching and Dialog Manager	122
5.12	The interaction between TGMAT classes and Target Group Database	126
5.13	Actions of the TGMAT main method	127
5.14	Selecting the node pair with <i>sv_max</i> from all node pairs built for profile node “Musik”	133
5.15	DM tasks and information packages passed between DM and other components	135
5.16	Interaction between DM classes, DM resources, and configuration settings . . .	137
5.17	Sample Question Specification	138
5.18	Constructing ISpecs for node pairs and inserting them into the question queue .	151
6.1	Samples of personal interests entered by users freely in different scenarios	160
6.2	LINGUINI test interface with starting question	172
6.3	Question and answer windows (top) and course of the dialog (bottom)	172
6.4	Question 30	174
6.5	Question 31 (A1, A2)	175
6.6	Question 31 (A3, A4)	175
6.7	Question 22	176
6.8	Question 26	177
6.9	Question 25	177
6.10	Question 11	179
6.11	Question 1, Question 12	179
6.12	Question 8	180
6.13	Question 9, Question 10	180
6.14	Question 4, Question 5	181
6.15	Question 14, Question 15	182
6.16	Correlations between user judgment, dialog length, and profile errors due to GermaNet and other system components	187
7.1	An example of a virtual character	195
7.2	Visualizing the complex profile representation for the user	197
B.1	Outlier values	218

List of Tables

- 4.1 Attributes that may or must occur in interest nodes 96
- 5.1 Semantic class tags for attribute values 104
- 5.2 Data structure representing a profile-TG-node pair 127
- 5.3 Similarity relations and relation values 130

List of Acronyms

ADJA	Adjektiv, attributiv (pre-nominal adjective)
ADJD	Adjektiv, adverbial (adverbial adjective)
ADV	Adverb
API	Application Programming Interface
APPR	Preposition
APPRART	Preposition with article
ART	Article (determiner)
CARD	Cardinal number
DIS	Dislike
DM	Dialog Manager
DOM	Document Object Model
HMM	Hidden Markov Model
HUM	Human being
IC	Information content
IRC	Internet Relay Chat
ISpec	Interaction Specification
IT	Information Technology
LA	Language Analysis
LINGUINI	L INGUistic I Nterest I dentification
LOC	Location
MD	Matching Degree
NC	Nominal chunk
NE	Named entity (proper name)
NER	Named Entity Recognition
NLG	Natural Language Generation
NLP	Natural Language Processing
NN	Nomen (noun)
PART	Participation mode
PC	Prepositional chunk
PDA	Personal Digital Assistant
PM	Profile Manager
PRS	Personalized Recommender System
P3P	Privacy Preferences Project
QSpec	Question Specification
RATE	Rating
SMS	Short Message Service
STTS	Stuttgart-Tübingen-Tagset
Synset	Synonym set
TEMP	Temporal expression
TG	Target Group
TGMAT	Target Group Matching

TGMOD	Target Group Model
UI	User Interface
UM	User Modeling
VC	Verbal chunk
VVFIN	Vollverb, finit (finite main verb)
VVPP	Vollverb, Partizip Perfekt (participle)
WSD	Word Sense Disambiguation
W3C	World Wide Web Consortium
XML	Extensible Markup Language

1 Introduction

1.1 Background

In this section we introduce the most important concepts, technologies, and research areas which have influenced and formed the basis for the work presented in this thesis. Our aim is to give a concise, introductory overview here. Most of the topics presented will be discussed in detail in Chapter 2 and 3.

1.1.1 Adaptive Systems and Personalized Recommender Systems

With the growing number of competing online vendors and information sources on the Internet the topic of *personalization* (cf. Section 2.1.1.2) becomes increasingly important. In order to make Websites more attractive, vendors do not merely aim at adapting the appearance of their platform interfaces to the customer's personal preferences, but also offer a variety of information items, services, and products which are tailored to the customer's needs.

Software systems which integrate personalization techniques and are hence able to adapt to the customer's preferences are called *adaptive systems* (cf. Section 2.1.1.1). These systems are capable of many different kinds of adaptation and research in this area is associated with a number of labels, such as user modeling, adaptive user interfaces, adaptive hypermedia, or personal learning assistants [Jameson 2001].

However, it is not always easy for customers to know exactly what their preferences are and what they are looking for considering the large number of choices available on the Internet. *Recommender systems* (cf. Section 2.1.3) may help users find and select items (e.g. books, movies, restaurants, documents, events, travel options, household appliances). Simple, non-personalized recommender systems, however, may suffer from the problem that a specific item that the system recommends does not fit each customer equally well. Thompson et al. [2004], for instance, therefore emphasize the need for *personalized recommender systems*. Personalized recommender systems are a specific type of adaptive system. They acquire information about the customer's preferences first and on this basis are able to make a more personal choice among recommendable items.

1.1.2 User Models

Before adaptive systems can offer products, services, or recommendations in a personalized way, they need to acquire personal information about the user. Such information is typically represented in *user models* (cf. Section 2.1.1.1). The scientific field investigating user models, acquisition techniques, and their application is called *user modeling* (cf. Section 2.1). Modeling the users of software systems, their preferences, intentions, plans, and beliefs has long been a topic in computer science. Early approaches came from the fields of human computer

interaction and artificial intelligence. With the emergence of E-Commerce a new view on user modeling has developed.

User models may contain different types of information, such as demographic user information (e.g. name, age, gender), interests, product preferences, ratings, intentions and goals, etc. (cf. Section 2.1.1.4). The models are typically categorized into explicit and implicit ones [Brusilovsky and Maybury 2002], according to their representation and acquisition technique. On the one hand, user models can be represented *explicitly*, i.e. an individual *profile* is stored for each user, which she usually can also see and modify. On the other hand, user models can be represented *implicitly*, i.e. they are typically hidden from the user. The two types of models also differ in the way in which they are acquired, as we will see in the next section. Moreover, user models can represent information about individual users or groups. *Group models* typically represent information which characterizes user stereotypes or target groups (cf. Section 2.1.1.4). The advantage of group models is that only little needs to be known about a user to assign her to a specific group and that the rest of the information can be inferred from the group model. The drawback is that the result will always be rather general user information which does not very well match the user in her individuality. Finally, user modeling approaches differ with respect to the information *life span*. Some approaches keep the acquired user information only for a single interaction, whereas others acquire long-term information which is applicable for multiple sessions with this user (cf. Section 2.1.1.4).

1.1.3 Acquiring Information about Users

Similar to the classification of user models, acquisition techniques are divided into the two main categories of implicit and explicit user profiling. These techniques require different levels of user involvement.

Implicit approaches acquire user information mainly by observation and do not ask the user for information directly. They analyze navigation actions users perform on a Website and take into account how long they have watched a particular page, which links they have followed, etc. On this basis, automatic inferences are made about user preferences. Another possibility is to infer user preferences from products users have bought. Implicit approaches are employed in most commercial user modeling applications [Kobsa 2001]. Their main drawback is, however, that implicit information is affected by a certain degree of unreliability. Moreover, it makes users lose control over the system's knowledge about them and the way this knowledge is used [Bauer 2004].

Explicit approaches, on the other hand, require the active involvement of users, who tell their preferences directly to the system. In most explicit approaches, users have to fill in an online questionnaire, which tries to elicit their preferences. In other cases, users are asked to rate products they have bought or information items they have seen. They may also conduct a natural language dialog with the system about their wishes, goals, and intentions. Although this directly entered information has the potential of being more reliable, the problem is that it requires a certain effort from the user, who has to fill in a large form or talk with the system in a lengthy dialog. As a consequence, several applications employ both implicit and explicit user models in order to overcome their individual drawbacks.

The main challenge for adaptive systems is to acquire reliable profile information without bothering the user too much. If users get overstrained, they may easily switch to another vendor or application, where less effort is required. A possible solution is to make the benefits clear to the users, for example, by directly rewarding them with an adequate and interesting product recommendation.

Acquisition methods are often tailored to specific applications and in general do not fit all types of applications equally well. Online questionnaires, for instance, which are a widespread means for acquiring explicit user information for Internet applications, are not suitable

in a mobile scenario where a user wants to access the Internet and get recommendations while driving a car. In most online applications, user information is acquired by means of typed input. Alternative input forms, such as speech dialog, are less explored. However, new application scenarios, in particular in the mobile context, require also investigating and developing new acquisition techniques.

1.1.4 Acquiring Information and Generating Recommendations in a Mobile Application Scenario

An especially interesting application field for personalization and in particular recommender systems are mobile scenarios (cf. Section 2.1.1.2). A mobile application has to handle changing contextual information, such as location, identity of users, time, and environment (cf. [Groh 2005]). Recommendations generated by a mobile application should be adapted to what is relevant for the actual situation of the user.

During the last decade, information technology for vehicles has achieved considerable attention. Today, telematic systems are standard equipment in high-end cars and enable, for instance, access to the navigation system, traffic advice, climate control, or entertainment (e.g. CD, satellite radio, MP3) [Lavie et al. 2005]. Since drivers of cars differ in terms of age, cognitive abilities, skills, computer experience, it would be desirable that the telematic system adapts to each user's needs [Lavie et al. 2005].

Dialogs in which driver and system communicate by means of spoken natural language are an important research topic in the area of in-vehicle information technology. Concurrent projects investigate, for example, how such speech dialogs can be used to enable access to various information sources, e.g. email, news, calendar, and Internet, but also to travel or hotel booking or to recommender systems. However, a robust speech dialog interaction is still difficult to realize (cf. Sections 3.2 and 3.7). It has also been investigated only rudimentarily how recommendations can be obtained in a moving vehicle and which contextual information should be taken into account in which manner.

We would like to illustrate with two sample scenarios why it is desirable to obtain personalized recommendations in a mobile context and how the above mentioned technologies can be integrated into a common application scenario.

Scenario 1 (vehicle)

A user could not plan her trip in advance or she has changed her route spontaneously and is now driving on a highway towards her travel goal. The long time spent with traveling on the highway could be used to elicit some personal preferences via speech dialog. In turn, the user could then get sightseeing and event recommendations about the travel goal via speech output.

User preferences could either be stored locally in the vehicle, on a chip the user carries with her (cf. Kobsa [2001]), or on a remote server which the user can access later to see and modify her profile. For obtaining recommendations, a recommender system could be located on a remote server. The recommender system has access to sightseeing and event information for specific regions, large cities, etc. which it can compare with the user preferences.

Scenario 2 (pedestrian)

The user walks through a city and wants to get recommendations about attractions, places to visit, events, leisure activities, etc. In this situation, she could use her mobile phone, for instance, to enter some personal information. Recommendations could then be generated for her again via speech output.

Here also, recommendations could be generated by a server which is contacted through a mobile phone connection. User preferences might be stored either on the remote server or on the mobile phone.

The approach to acquiring explicit user information we have developed is applicable not only for the two scenarios presented above, but for various types of situations and applications, e.g. for E-Commerce Websites, where a typed dialog interaction is also possible. In this thesis, we use Scenario 1 (the vehicle) only as an example for illustrating our solution. In the next section we motivate our approach and describe the problems which have to be solved.

1.2 Motivation and Problem Description

The scenario of a driver accessing online services poses some interesting questions and problems. A central question is, for instance, how the driver can interact with the computer system without being distracted too much and without exceeding safety limits. Information technology applied in vehicles must not increase driver distraction or cause safety problems but should ideally rather lower distraction [Lavie et al. 2005]. Therefore, visualizing extensive amounts of information (e.g. questionnaires or recommendations) for the driver on a screen does not seem to be a good solution. Minker et al. [2003] suggest employing speech recognition technology in combination with a spoken language dialog system. This does not only increase road safety by enabling the driver to continuously watch the traffic but is also considered the most natural and appropriate way of communication [Minker et al. 2003]. Speech interaction moreover solves the problem of users providing information (e.g. for their user model) which they normally would have to enter via keyboard, which is not suitable in a vehicle.

These specific interaction modalities do not only affect the user interface of a recommender system operated in a vehicle. They also influence the applied profiling method and even the user model itself. In the course of this thesis, we will explain in detail why we decided to acquire and model profile information explicitly (cf. Section 2.2). The main reason for this decision is that we wanted to take a profiling approach which leads to reliable profile information, since this is an important prerequisite for generating adequate, high-quality recommendations. We have already mentioned drawbacks of implicit profiling in this respect and will further discuss them in Section 2.2.2.1. In the vehicle, we also do not have the possibility to observe user actions such as browsing Websites and products, from which we could derive user preferences automatically. We noticed, however, that most explicit profiling approaches are not personalized, but rather use standardized questions for each user. They furthermore require a lot of effort from the user and only a small amount of information can be acquired with them, which makes them less attractive in practice. We think that improved methods for explicit profiling are not only desirable for the vehicle scenario, but also for adaptive systems applied in other areas, e.g. in online stores of Internet vendors.

Another problem of many profiling methods, which we want to address with our approach, is that they merely offer a closed set of possible answers. Since only a few items can be presented to the user in this way, these items cannot be highly personalized and will not match many users very well. A solution is to allow free user input, i.e. users may describe a preference in their own words rather than selecting from a predefined and limited answer set. Zadrozny et al. [2000] view natural language processing as a compelling technology which enables personalization exactly for this reason. The benefits of natural language have already been investigated for non-mobile adaptive systems. Natural language user profiling applied in recommender systems, for instance, facilitates rich feedback in a way that is natural for users and takes into account their individual vocabulary preferences [Wärnestål 2005]. Allowing

natural language user input, however, requires much more complex capabilities for interpreting this input than are necessary for the closed answer set alternative. User answers have to be analyzed and a solution has to be found for storing the information in the profile in a useful way.

If the system is supposed to engage in a natural language interaction with the user, it needs a device which controls this interaction. This is typically accomplished by a (speech) dialog system (cf. Section 3.7). There are many different types of dialog systems and the choice of which one to apply depends on a variety of factors. Important characteristics which may influence the decision process are the following:

- **Application domain:** How does the application domain influence the dialog? What is specific about dialogs conducted in this domain as opposed to other domains? Which domain-related terminology needs to be covered by the system?
- **Usage environment:** In which environment will the system be used? Does the dialog system have to have special capabilities to cope with environmental restrictions (e.g. in a vehicle the dialog system has to be more robust with respect to background noise than if the dialog were conducted with a desktop computer)?
- **Dialog goal:** What goal is supposed to be achieved with the dialog? What is the specific task the user may accomplish with the dialog interaction?
- **Depth of understanding:** Which depth of linguistic interpretation (e.g. shallow or deep analysis) is required from the system?
- **User adaptability:** Considering the dialog goal, is it sufficient that the same dialog is conducted for each user, or is it necessary or desirable that the dialog adapts to each user individually? In which respects should the dialog be adaptive (e.g. adapt the topics of interest, adapt to the user's expertise and level of knowledge)?
- **Initiative:** Is it better to employ a fixed-initiative (e.g. initiative is only taken by the system) or a mixed-initiative approach (both system and user may take the initiative)? Which strategy is best for achieving the dialog goal?
- **Dialog duration:** Which overall dialog duration is most suitable and still acceptable for achieving the dialog goal?
- **Configurability:** Will the dialog system be used by different applications? Should application designers be able to configure the dialog in order to adapt it to the specific needs of their applications? Which parameters should be configurable?

Our sample application scenario shows the following characteristics with respect to these aspects. The **goal of our dialog** is to acquire as many personal interests of users as they are willing to give and to store them in semantically structured profiles, which reflect relations between interests and related topics. The profile information is supposed to be usable for generating personalized recommendations on the basis of these interests. Since we allow free user answers, the dialog must be able to handle an **unrestricted vocabulary**. The utterances produced by the system during the dialog have to be tailored to the **domain of acquiring personal interests**. Our sample **usage environment** is the vehicle, i.e. the dialog system must be robust in understanding continuous, spontaneous speech which competes with a certain amount of background noise. For various reasons which we will discuss in the course of this thesis, the user input allowed should be structurally restricted, i.e. the user should enter words or phrases rather than one or several complex sentences. This has the advantage that the necessary **depth of understanding** can be accomplished by a shallow linguistic analysis. Furthermore, we assume that a profiling system is most valuable if it can be reused for different

applications. Therefore, we consider **configurability** an important aspect. Application designers should be able to configure, for instance, dialog length with respect to their needs and the expected user behavior. Since the level of detail of the profile will increase with the **dialog duration**, it is not reasonable to aim at the shortest dialog possible, however. The decision as to whether the dialog should be **user-adaptive** and which **initiative strategy** should be taken, is not forced into a specific direction by the sample scenario. We found that for our dialog goal, the strategy in which only the system takes the initiative is sufficient. Instead, we found it more important to put considerable effort into the dialog's adaptability. In the remainder of this section, we explain why the dialog's capability to adapt to free user input supports our dialog goal.

Without regard to the type of dialog system chosen, the main aim for the dialog designer should be to avoid causing the user to get bored or overstrained and break off the interaction before the dialog goal is reached. Users' interest and cooperation can be influenced, for example, by the time they have at their disposal, by more or less direct benefits they perceive from conducting the dialog, by the degree of urgency with which they want to accomplish a task or obtain information, or by the system's linguistic capabilities and competence. There is another important aspect which may influence the user's cooperation and satisfaction positively: giving more freedom to the user with respect to input possibilities. We already mentioned some advantages of allowing free user answers in our application scenario. McNee et al. [2003] argue for the benefits of "allowing users to more fully express their interests across the entire universe of items". They show that user loyalty increases and that users even accept a longer profiling process (e.g. rating movies), if they are able to actively influence this process by choosing the items to rate themselves. This has encouraged us to believe that a similar effect may take place, if we allow free user answers and hence enable the user to influence the dialog, which adapts automatically to the user's topics. Conducting a dialog that is perceived as personalized may well be more entertaining than a dialog that is not adaptive and only allows for predefined answer choices. We also expect that the motivation to cooperate is high in the sample application scenario, because users have a lot of time at their disposal (e.g. on a long highway trip) which they cannot use for many other things. Moreover, their motivation to obtain event and sightseeing recommendations is probably also high, because they are about to reach their travel goal. If they only have a couple of hours to spend there, they will probably not want to waste time with searching for a tourist information centre.

1.3 Goals of this Thesis

The **overall goal** of this thesis was to develop a generally applicable method for acquiring user profile information, which can be illustrated by means of the above described Scenario 1 (cf. Section 1.1.4). This goal can be divided into **two main tasks**:

1. The development of a **natural language dialog approach** which is
 - suitable for acquiring personal interests of users
 - able to analyze and react to not predefined user input
 - adaptive by focusing on the individual user's interests and related topics
 - usable in various application scenarios with either speech or typed input/output interfaces

2. The development of a **profiling approach** which

- acquires, stores, and maintains user information explicitly
- represents user interests and semantic relations between interests in a structured way
- enables long-term maintenance and exchange of user information among applications

Both parts constitute the **main contribution** of this thesis. We have designed the LINGUINI (“**LINGU**istic **IN**terest **Id**entification”) approach, which integrates both parts, and implemented it as a prototype system.

The adaptive natural language dialog approach we have developed depends on a semantic knowledge representation of user information, which allows us to find related interests. The dialog design and strategy, moreover, had to be suitable for acquiring personal interests. Since there are no existing profiling systems which either combine these two key characteristics or could be adapted for our purposes with justifiable effort, we decided to develop most of the system from scratch. The only **external technologies and models** which are used in the LINGUINI profiling approach are a natural language processing tool for part-of-speech tagging and chunking, a lexical-semantic network, and a sociological target group model.

There are some **aspects which could not be implemented or investigated in detail** in this thesis. It was not a goal of this work to also develop a recommender system for sightseeing and event information or to investigate the performance of such an application in connection with LINGUINI. Furthermore, we will clarify theoretically, why our approach enables exchangeability of profile information between applications. It would exceed the scope of this thesis, however, to also investigate this subject empirically here. Finally, considerable effort was spent in not only simulating but actually implementing the various functions of the two central tasks described above in the prototype system. These are realized in the main components Dialog Manager, Profile Manager, Language Analysis, and Target Group Matching. In order to be able to develop these extensive components in full detail, we decided to only realize a simple Question Generation component and a typed input/output User Interface instead of a speech interface. The integration of a more elaborate natural language generation component and a speech interface is discussed theoretically in Sections 3.2 and 3.8, however.

1.4 Methodology and Structure of the Thesis

1.4.1 Methodology

This thesis follows a constructivist research approach. Constructivism first requires specifying a **problem** which could not be solved by existing approaches until now. The next step is the **construction of a solution**, whereby it is crucial to take into account existing research and defend the choice of elements for the solution accordingly.

The problem solved in this thesis is the development of an explicit profiling method which is applicable in the mobile recommendation scenario sketched above. In particular, personal interests are supposed to be acquired by means of an adaptive natural language dialog. The construction of a solution has been accomplished as follows. Prior to our work, we conducted an empirical user study in order to find out how users would formulate their interests in natural language. On this basis, we designed our profiling approach with the characteristics described in the previous section. We implemented a prototype system and evaluated this prototype by means of a second user study. In this study, users conducted the profiling dialog with the system and evaluated its adaptability and quality by means of a questionnaire. The study was followed by an objective profile quality assessment.

1.4.2 Structure

The remainder of the thesis is structured as follows:

Chapter 2 presents the fundamentals for this work with respect to user modeling. We first learn about different user modeling approaches and see how user models can be applied (Section 2.1). We argue for our user model on the basis of existing approaches and introduce personalized recommender systems. After this, methods for acquiring user information are discussed (Section 2.2). At the end of this section, we argue for the profile acquisition approach of this thesis.

Chapter 3 describes fundamentals with respect to natural language processing. After introducing the topic of applying natural language processing for user profiling (Section 3.1), this chapter describes the natural language processing methods and resources that play a role for LINGUINI. It explains why they have been chosen and for which tasks they are used. These methods and resources are speech recognition and synthesis (Section 3.2), part-of-speech tagging (Section 3.3), syntactic analysis (Section 3.4), lexical-semantic networks (Section 3.5), word sense disambiguation (Section 3.6), speech dialog systems (Section 3.7), and natural language generation (Section 3.8).

Chapter 4 gives an overview of the profiling approach developed in this thesis. It starts with a summary of requirements (Section 4.1), which are derived from the observations made in Chapters 2 and 3. It then gives an overview of the dialog process from a user's and a system-internal point of view (Section 4.2). In Section 4.3, the LINGUINI system architecture is presented. The chapter concludes with the detailed presentation of our user model, which is central for the other system components (Section 4.4).

Chapter 5 describes in detail the main components of LINGUINI. The Language Analysis component is responsible for the analysis of the natural language input of the user (Section 5.1). The Profile Manager constructs user profiles and augments them with further information during the dialog (Section 5.2). The Target Group Matching calculates a most similar user group for the customer and by doing so supports the Dialog Manager in choosing a suitable next question (Section 5.3). The Dialog Manager is the central system component and controls the dialog interaction with the user (Section 5.4).

Chapter 6 discusses the implementation of the solution and the results of the system evaluation. Section 6.1 presents the results of the empirical study we have conducted prior to the development of our solution in order to find out how users would formulate their interests. Section 6.2 describes implementation aspects of the LINGUINI prototype system. In Section 6.3, we present the final evaluation of our prototype, which consists of two parts: a user study about dialog adaptability, user acceptance, and development status and a succeeding quality and coverage assessment of the created profiles.

The thesis concludes in **Chapter 7** with a summary of the main contributions and results and prospects for future work.

2 Modeling and Acquisition of User Profiles

The aim of this chapter is to give an overview of existing approaches and to introduce fundamental concepts and methods in the area of user modeling. We will first discuss user models and their application and will see how these user model types relate to our approach (Section 2.1). Afterwards, we will describe existing profile acquisition techniques, which are compared with the methods applied in LINGUINI (Section 2.2). Both Sections contain a detailed motivation of and requirements for our own work. Because of the variety of existing approaches we consider it most helpful for the reader to be able to compare the characteristics of these approaches with those of LINGUINI directly in connection with their presentation.

2.1 User Modeling and Applications

2.1.1 A Survey of User Modeling Approaches

2.1.1.1 User Models and Adaptive Systems

The idea of *user modeling (UM)* is to represent user preferences as a computer model which can be interpreted by a system to adapt its actions, services, layout, etc. to the individual user. Systems which adapt to the user's preferences have already been under investigation for a long time [Rich 1979, Langley 1997]. User-adaptive systems often occur under different names, such as *adaptive interfaces*, *user modeling systems*, or *(adaptive) intelligent agents* (cf. [Jameson 2003]). They have to be distinguished from *adaptable systems*, which users can tailor to their own preferences, for instance, by modifying the appearance of the user interface [Jameson 2003]. We will follow the definition of *user-adaptive systems* given in [Thompson et al. 2004] in this thesis:

Definition 2.1 (Adaptive System, User-Adaptive System [Thompson et al. 2004]) Personalized, *user-adaptive systems* obtain preferences from their interactions with users, keep summaries of these preferences in a user model, and utilize this model to generate customized information or behavior. The goal of this customization is to increase the quality and appropriateness of both the interaction and the result(s) generated for each user.

Definitions of a *user model* found in literature often depend on the application type employing the model. This is illustrated by the following definition taken from the application field of natural language dialog systems:

“A *user model* is a knowledge source in a natural language dialog system which contains explicit assumptions on all aspects of the user that may be relevant to the dialog behavior of the system.” [Kobsa and Wahlster 1989]

We generalize this definition to an application-independent notion of *user model* in the following. Moreover, we define the terms *profile model* and *profile instance* as used in our approach. Figure 2.1 illustrates the relationship between these concepts.

Definition 2.2 (User Model) With the term *user model* we refer to a system-internal, formal representation of user properties and preferences, which are used for personalizing system actions and services. We use this term as a superordinate concept covering all types of user representation in adaptive systems.

Definition 2.3 (Profile Model) With the expression *profile model* we refer to the particular kind of user model developed in this thesis. The profile model defines how a *profile instance* can be structured, i.e. it is a model for generating profile instances in LINGUINI.

Definition 2.4 (Profile Instance, Profile) With *profile instance* (alternatively: *(user) profile*) we refer to a particular profile representation that has been created for an individual user and has been *instantiated* with her personal preferences. In our case, this profile instance is stored as explicit graph structure in a database.

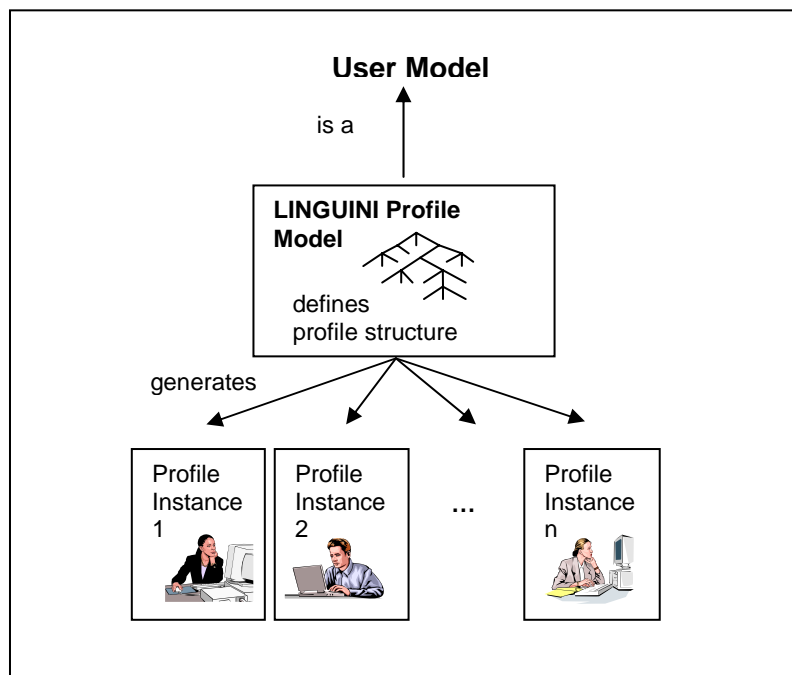


Figure 2.1: Relations between user model, profile model, and profile instances

2.1.1.2 Technological Areas in which User Models Are Developed and Applied

UM has been investigated by various disciplines in the field of computer science, which work with user models for different purposes. An early interest in UM has been shown in the field of Artificial Intelligence. Today, UM is mainly used for improving human computer interaction and for adaptive systems, particularly in the area of E-Commerce. The number of profile-based systems offering adaptive services via the Internet is constantly growing. In this section we describe these areas and briefly sketch the chronological development of UM.

Early Approaches in Artificial Intelligence

UM has its origins in the late 1970s. According to [Kobsa 2001], there was no separation between UM and the application itself in these early systems and reusability of the profiles was not yet considered.

One of the earliest approaches to developing an application-independent UM system was GUMS [Finin and Drager 1986, Finin 1989]. GUMS was designed to allow the application designer to specify her own user model representing user stereotypes. It hence set the basis for systems that could be filled with application-specific knowledge. The widely-used notion of *user modeling shell systems* (short: *shell systems*) has been introduced by [Kobsa 1990] for profiling systems that use a modular software architecture to achieve modifiability and reusability and that can be adapted to the needs of the application. In the 1990s, major shell systems were developed, for instance, BGP-MS, UMT, DOPPELGÄNGER, or TAGUS (cf. [Kobsa 2001]).

Early approaches to UM were strongly influenced by research conducted in the areas of Artificial Intelligence, natural language dialog [Kobsa and Wahlster 1989, Zukerman and Litman 2001], and intelligent tutoring (e.g. [Kass 1989]). According to [Zukerman and Litman 2001], early user models represented sophisticatedly modeled dimensions, such as user goals, plans, beliefs, capabilities, or attitudes. These user models unfortunately only worked for specific examples, whereas nowadays, approaches are favored that use a shallower knowledge representation, behave more robustly, and set focus on automatic acquisition of the profile. The early, rather complex techniques became less attractive when in the mid-nineties adaptive systems developed towards a more simplistic approach to UM. Two of these more simplistic approaches are user-adaptive learning environments and user-tailored Websites (cf. [Kobsa 2001]).

In this early phase, there were also approaches to improving natural language dialog systems by means of user models. For most of these systems, the main purpose of a user model was to make the system adapt the dialog to the user's assumed goals, plans, or beliefs. The idea of improving the dialog by means of user models is still investigated today. However, more systems have evolved recently which use natural language dialog for collecting profile information that can be used for various adaptive purposes, not only for improving the dialog interaction itself. A detailed survey about the early UM approaches, especially for the use in dialog systems, is given by [Kobsa and Wahlster 1989].

Personalization and E-Commerce

In the late 1990s, UM became an increasingly popular topic in E-Commerce, and *personalization* has been identified as an important issue for the development of Web applications. [Tseng and Piller 2003] define personalization as follows:

Definition 2.5 (Personalization [Tseng and Piller 2003]) Personalization in general is about selecting or filtering information objects for an individual by using information about the individual (the customer profile) and then negotiating the selection with the individual.

This definition is very close to Definition 2.1 for adaptive systems. In fact, the concept of personalization is inherent to most adaptive systems. The basic idea of personalization in E-Commerce is to learn about the customers' preferences and to use this information to tailor online service or information offers to their needs.

Although UM became an urgent requirement with the emergence of personalization, the UM shell systems mentioned above were not widely used in E-Commerce. This was mainly due to the complexity of their user models. However, some of their central ideas (e.g. the client-server architecture and the use of stereotypes) were further applied and developed in

commercial systems [Kobsa 2001]. For a detailed account of commercial user modeling servers developed in the late 1990s see [Fink and Kobsa 2000].

In the following, we describe three topics related to UM and personalization in E-Commerce which are currently under intense investigation:

Personalized intelligent agents and chat bots: In E-Commerce, investigations often center on online sales dialogs and the problem of finding out the needs of potential buyers. There is also the problem of addressing inexperienced Web users in a natural, intuitive way to offer new products, inform about new site features, direct them to sub-pages of interest, etc. *Animated intelligent agents* (also: *artificial* or *virtual agents*) (also cf. Section 7.2.1), which conduct a natural language dialog with customers, may be used to solve these tasks and hence increase the customer's loyalty to the Website. However, virtual agents can become arbitrarily complex, employing various technologies from Artificial Intelligence to make them resemble intelligent beings. Most Internet vendors are looking for more inexpensive, faster software solutions. For this reason, they often apply *chat robots* (short: *chat bots*) (cf. Section 2.2.3.1). Chat bots are rather basic software programs that can be placed in chat applications and that simulate the communicative behavior of human chat users. The main drawback of chat bots is that their language understanding capabilities are limited and mostly restricted to pattern matching, such that they typically do not reach the standard of state-of-the-art dialog systems or virtual agents. However, with considerably less technical and financial effort as compared to complex virtual agents, chat bots are easy to apply to different Websites (e.g. by using the Lingubot Creator™ by Kiwilogic, www.kiwilogic.com). Both virtual agents and chat bots can acquire and employ user models to adapt to the user's preferences [Pandzic 2001].

Mass customization: With the beginning of the 21st century, another aspect of personalization and Internet comes into play that also requires information about user preferences: the design, presentation, and manufacturing of individualized products. The area of *mass customization* [Pine 1993, Piller 2001] investigates these topics. With the beginning of industrialization and the advent of mass products, customers had to adapt to the purchased products. Later, customers had the choice between different product variants, and now manufacturers are trying to adapt their products to a single person. Offering mass-customized products via Internet, however, requires capturing the individual taste and needs of a customer exactly by means of appropriate profiling techniques. Moreover, the software specification of the individual product wish can become very complex and may overstrain the customer, especially for complicated mechatronic products. Customers therefore need to be supported by software tools which help to reduce this complexity. A solution is, for instance, to support customers with new types of specification tools, which integrate various personalization techniques [Leckner et al. 2004]. A specific personalization technique is, for example, the automatic generation of product recommendations during the specification process as described in [Stegmann et al. 2003, Renneberg et al. 2004].

Recommender systems: A technology which has gained increasing attention during the last years, not only in the area of mass customization, is recommender systems applied on the Internet (e.g. [Schafer et al. 2000]). The task of recommender systems is to offer selected information (e.g. news, Web articles, ads, events, sightseeing possibilities, restaurants) or products (e.g. books, gifts, computers, household appliances, travel options) that may be of interest to the visitor of a Website. For tailoring recommendations to the individual user, many recommender systems collect information about the customer implicitly or explicitly. These personalized recommender systems will be discussed in detail in Section 2.1.3.

Mobile Applications and Ubiquitous Computing

Personalization and UM also play an increasingly important role in applications that can be accessed not only from a desktop computer, but from any possible location. The research field which investigates this idea is *ubiquitous computing* (e.g. [Weiser 1993]). In ubiquitous com-

puting, the vision is that the user does not even need to carry a computing device (e.g. a PDA) any more, because information is accessible everywhere and information displays are integrated into the everyday world [Weiser 1993]. In this envisioned setting it is desirable that users can also access their specific personal information environment anywhere [Kobsa 2001]. Prerequisite for ubiquitous computing is the availability of wireless networks between multiple omnipresent devices. This vision challenges new types of applications, which can cope with the requirements of these complex networks of distributed and mobile computing devices.

Today, there are already a variety of approaches that investigate user preferences in mobile or ubiquitous scenarios. We will mention a few examples of recent activities to illustrate this. A number of research projects conducted by major car manufacturers aim at the adaptation of online services accessible in the car to the individual driver's needs (cf. Section 3.7.2). Another possible application is so-called wireless or mobile recommender systems [Jameson et al. 2002]. Requirements for profile-based, adaptive interfaces for ubiquitous Web access are investigated by Billsus et al. [2002]. The project COSMOS ("Community Online Services and Mobile Solutions", www.cosmos-community.org), conducted at the Technical University of Munich together with the mobile phone company O2 Germany AG, investigates how mobile virtual communities can be supported technologically and by generic concepts for services. In [Groh 2005], an approach is presented that uses information about personal interests to form ad-hoc groups for common activities in mobile communities.

Technologically, the realization of mobile user profiling poses some problems, however. Since most existing profiling servers require the network connection to persist constantly, new forms of profiling seem to be necessary for ubiquitous scenarios. Kobsa [2001] proposes to investigate the use of a *user model agent* (instead of a conventional user model) which resides on the server side and is replicated at the beginning of each interaction. Alternatively, Kobsa [2001] suggests that the agent might be located on a chip, which is integrated into a personal object the user carries (e.g. plastic card, wristwatch, jewelry) as described for the AVANTI system in [Fink et al. 1997]. AVANTI is an adaptive system for providing tourist information (cf. Section 2.1.3.2).

As this thesis presents a profiling approach that can be used for generating in-vehicle recommendations, we will discuss further aspects of mobile user profiling at several points throughout the remainder of this chapter.

2.1.1.3 Issues in User Modeling

Issues and problems related to particular UM approaches will be discussed throughout the course of Chapter 2. In this section (Section 2.1.1.3), however, we summarize some key issues, which *all* (i.e. mobile or other) UM approaches have to address in some way.

Privacy

A well-known issue in UM is privacy. Many users are concerned about software systems storing information about them. The fear of misuse inhibits the users' willingness to provide insight into their wishes and needs, even if this is connected with a direct gain (e.g. a product recommendation). [Cranor 2003] outlines that privacy concerns can be reduced if adaptive systems, for instance, allow for pseudonymous interactions, data storage on the client side, and interfaces where users are able to control the collection and use of their personal information. One of the most important privacy-enabling movements is the Privacy Preferences Project (P3P) developed by the World Wide Web Consortium (W3C). It investigates how users can be given more control over personal information that is acquired about them when they visit Websites. The P3P privacy policy is emerging as an industrial standard and is used by

many Websites today. The idea is that Websites make it clearly visible if they use the P3P policy, store this policy in a standardized and understandable form and give users the freedom to act on what they see. The P3P-enabled Privacy Bird [Byers et al. 2004] is a tool which helps users to stay informed about the privacy policy of Websites they visit. Besides clearly stated privacy policies, clearly manageable access rights to user profiles can help to increase the user's trust into the system [Wörndl 2003].

Transparency and Degree of User Involvement

Closely connected with privacy is the topic of transparency (e.g. [Bauer 2004]). With *transparency* we refer to the circumstance that personal information collected is made visible to users and that no data collection happens that they do not know of or cannot control. Besides having to inform users about the collection of data, which is legally mandatory [Wörndl 2003], any adaptive system has to address the question, how much of the information collected about users should be made visible to them and in which format. Should they be able to modify or even delete parts or all of the information? Will this increase or reduce trust in the application?

This also raises the question of how much users should be involved into the process of collecting profile information at all. This decision has to be handled carefully, since to completely exclude users from the profiling process may result in a direct loss in profile quality. Merely observing user behavior can lead to misinterpretations, which would rather increase mistrust when shown to the user. Furthermore, the information is often stored in system-internal representations which are difficult to visualize for users. Therefore, we argue for profiling methods that acquire information directly from the user as discussed in Section 2.2.2.2, or at least for a combination of both. This requires a stronger user involvement on the one hand, but also has a greater potential for realizing transparency and hence for increasing trust and cooperation on the user's side. We discuss the above mentioned problems connected with implicit profiling in detail in Section 2.2.2.1.

Reusability

Several authors (e.g. [Kobsa 2001, Koch 2002]) emphasize advantages of a centralized storage of user profiles and their accessibility by different applications. Reusability of the profile is the key idea here. Information is entered only once by the user and is stored on a central server. This is supposed to replace a number of tedious interactions, whereby the user has to enter the same information over and over again for different applications.

A question connected with reusability is about the contents and data representation which are most suitable for profile exchange between applications. There are various approaches aiming at the development of standardized profile formats. One of them is, for instance, the specification developed by CPExchange (Customer Profile Exchange, www.idealliance.org/cpexchange), an organization dedicated to developing an open standard to support the privacy-enabled global exchange of profile information. Such standards also make use of exchange-supporting representation formalisms as, for example, XML, which is also used for profile representation in this thesis.

Centralized profiling has many advantages, such as reusability, avoiding redundancy, a general approach to identification, authentication, and access control (e.g. [Fink and Kobsa 2000]). Nevertheless, disadvantages, such as the necessity of a network connection and potential central point of failure, also have to be taken into account [Kobsa 2001].

2.1.1.4 Types of User Models

On the background of a multitude of both research approaches and commercial UM approaches it is not surprising that user models differ considerably among applications. This section gives an overview of possible types of user models and differences in their contents and data representation. We follow Thompson et al. [2004] in distinguishing user models according to five characteristic criteria, presented in the following.

1. Individual vs. Stereotype Information:

User models can differ in that they represent individual users or stereotypes. In UM approaches based on *stereotypes* (e.g. [Rich 1979, Kobsa and Pohl 1995, Ardissono and Goy 2000]) customers are assigned to a mainly stereotype user model (e.g. “sports person”) that matches best their interests, preferences, attitudes, etc. The system makes this assignment on the basis of a certain amount of personal information that is known already (i.e. which the user has entered or which has been observed). Additional properties of the user are then inferred from the (mainly hand-crafted) stereotype model without further questioning. Stereotype approaches were especially popular in the early days of UM. [Kobsa 1990] reviews some of these early stereotype models.

Currently used stereotype models are typically based on psychological, sociological, or domain-specific knowledge. They are therefore closely related to so-called *target group models* (e.g. [Schulze 1992, Spellerberg and Berger-Schmitt 1998]) which are developed and used in sociology. The purpose of target group models is to classify the population of a country into groups, often called *milieus*, that show a common demographic situation or lifestyle, similar attitudes and personal interests. The attributes that characterize a milieu mostly have weights which help to determine how significant an attribute is for this specific milieu. Target group models (e.g. the Sinus-MilieusTM target group model by Sinus Sociovision, www.sinus-sociovision.de) also play an important role in marketing. A classic marketing strategy is to assign new customers to milieus which are associated with certain product preferences. After this assignment, milieu-directed advertisement and product marketing is possible. For a detailed comparison of Schulze’s [1992] target group model and the Sinus-MilieusTM model we refer to [Diaz-Bone 2004].

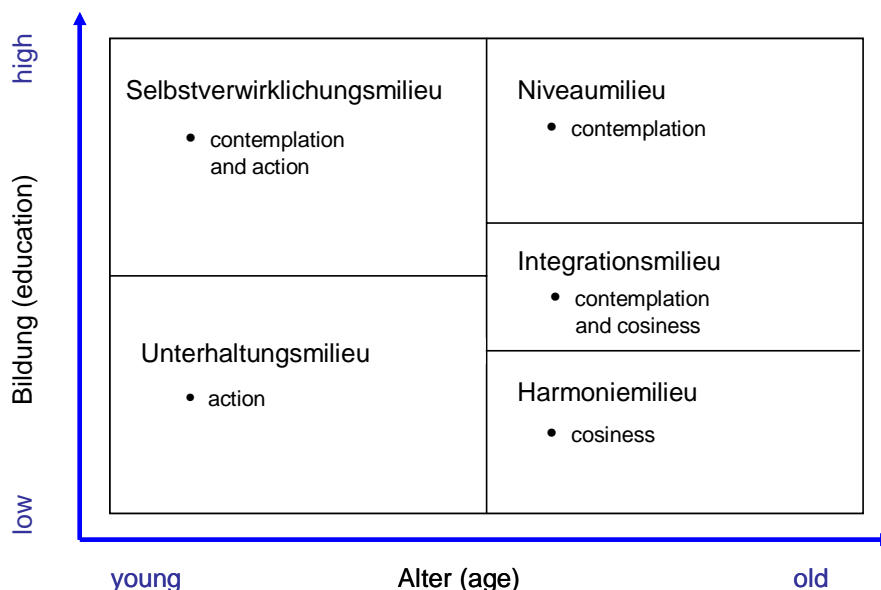


Figure 2.2: Target group model by Schulze [1992], extended with enjoyment attitudes

Figure 2.2 shows the sociological milieus assumed by Schulze [1992] in relation to education and age of the German population. Each milieu is characterized by a specific attitude towards enjoyment (“Genußschema” [Schulze 1992]) indicated below the milieu name. This target group model is also used in LINGUINI (cf. Sections 2.1.2.1 and 5.3).

The drawback of stereotypes or target group models is that they make generalizations which are not appropriate for each user alike. Furthermore, assuming that a stereotype or target group model matches a large number of individuals seems not to meet the current development towards personalization. Therefore, models which represent users more precisely and individually become increasingly important. With the term *individual user models* we refer to approaches that store information which has been acquired for this particular user explicitly or implicitly. Individual user models typically do not contain generalized information derived from stereotypes or group models. Most of the approaches we discuss in this chapter use individual user models.

2. Handcrafted vs. Learned Models:

User models can also be distinguished according to the way they are created: manually or automatically, for example by employing machine learning techniques. The properties of *learned models* often depend directly on the application which supplies the data for the learning. Such data can, for example, be questionnaires, ratings, or usage traces [Thompson et al. 2004]. An advantage of this approach is that if profile information changes over a longer period of time, these changes can be learned and an update of the model can be made automatically. The drawback is that, in general, a large amount of data, which can be used for training the learning algorithms, has to be available in the application from the very start.

Handcrafted models, on the other hand, are constructed by a profile designer. Since this process is independent of data already collected in the application, a handcrafted model can be used right from the beginning. A disadvantage is that this approach requires more manual effort, both for the initial construction and for subsequent changes or updates.

3. Differences in Informational Contents:

User models can represent different types of user-related information. This may be *simply structured information*, such as **personal information** (e.g. name, address, age, bank account, education), **user behavior** (e.g. previously watched, selected, or bought items), **preferences about item characteristics** (e.g. color, functions, price), or **ratings** (e.g. for books the user has read, movies she has seen, or bought products). Earlier approaches to UM also tried to represent more *complex structured information* such as **plans, beliefs, or goals** of users, but mainly with respect to some specific task users want to perform with the application (e.g. train booking). Further types of user information which may also have a complex semantic structure are, for example, **personal or professional interests, attitudes, and relations to other users**. As opposed to the task-bound information mentioned above, these interest types may be relevant for various tasks and applications.

Both simply and complex structured contents often depend on the application which uses the model. Most systems represent only a small part of the user information types mentioned above according to their specific needs. This implies that these specific models cannot be re-used easily for other applications. A solution are standards for user models (cf. Section 2.1.1.3), which comprise not only privacy aspects and format, but also aim at standardized and reusable contents.

4. Single Session vs. Long Term Models:

There are user models which exist only for the duration of a single session in which the user interacts with a system, whereas others have a multi-session lifetime. Most early approaches

to UM use single session profiles which are discarded once a session terminates. Today, many Internet vendors (e.g. the Amazon online bookstore, www.amazon.de) maintain profiles over a longer period of time during which a customer visits a Webstore repeatedly. There are two main reasons for this. First, the vendor wants to avoid having users switch to another Webstore. Having entered information for one vendor which can be reused during later sessions with this vendor is expected to increase customer loyalty. Secondly, collecting information over a longer period of time enables the vendor to draw inferences about customer behavior and interests, which can be used for improving online marketing and personalization services.

Long term models have to take into account that user interests may change over a period of time [Mertens and Höhl 1999]. Moreover, user preferences may vary according to different situations which occur during the use of applications. Solutions to this problem are to apply learning methods or to give profile access to users such that they can control the changes themselves.

Long term models also help to solve the *Cold Start Problem* (e.g. [Koch 2002, Schein et al. 2002]), which occurs if an application does not yet have the information needed for a specific personalization service. A solution for this problem – which helps to avoid asking the user to reenter information – is the *import of profiles*. In this approach, the user is supported in making information which has already been entered for some applications accessible for further applications [Koch and Möslin 2003]. Profiles for long-term use are stored centrally and can be accessed by different services and applications. [Koch 2002] argues that customers should remain the owners of their profile information in order to achieve a trustful relationship, which is the basis for the customer to reveal and store personal information.

5. Direct-Feedback vs. Unobtrusive Acquisition:

Thompson et al. [2004] also distinguish user models according to how they are acquired. In the *direct-feedback* approach (also called *explicit profiling*) users enter information about themselves or their preferences directly into the system. Another possibility is to acquire information *unobtrusively* (also called *implicit profiling*), i.e. the users' online behavior is observed and inferences about their preferences are made on this basis. In the direct-feedback approach, user information is not only acquired directly from the user but mostly also stored in an explicit user profile, for example in the form of attribute-value pairs. In the case of unobtrusive acquisition, the observed information is typically not represented explicitly. Instead, it is stored in a system-internal format, which in general cannot be seen or modified by the user, not to mention exchanged between applications (cf. [Mehta et al. 2005]). We will learn more about these acquisition techniques in Section 2.2.2.

2.1.2 The User Modeling Approach of this Thesis

On the background of the UM approaches described above, we will now argue for the type of user model developed and applied in this thesis. Our user model has to fulfill certain requirements imposed by the aim of acquiring highly individual information by means of a user-adaptive, natural language dialog. The **main requirements** for the model are:

- It has to be suitable for representing user interests entered in natural language, which show a more complex semantic structure than basic personal data (e.g. name, age).
- It has to be flexible enough to integrate user input which has not been predefined in the model before.
- It has to define and represent semantic relations between information items. This is necessary for making inferences about related topics while selecting the next question.

- It should be application-independent, i.e. representation and contents should be reusable by different applications

The few existing user models which are application-independent are not flexible enough to handle previously unknown user input. Moreover, most standardized explicit models only store personal information of a rather simple structure. This information can often be represented by a single attribute value. These models do not allow for more complex data structures which can be derived from natural language phrases or sentences. Profile hierarchies, which indicate how items semantically relate to each other, are mainly static and cannot be extended for new topics, if used at all.

For these reasons, we decided to develop a new user model for LINGUINI, henceforth called *Profile Model* (cf. Definition 2.3). Our Profile Model shows the subsequent characteristics and relates to the five criteria of Thompson et al. [2004] described above as follows:

1. The Profile Model represents **individual** (as opposed to stereotype) information. We use a target group model in addition for controlling the dialog flow (but not as a user model).
2. The underlying model is **hand-crafted**, but the individual profile instances are created automatically on the basis of a lexical-semantic network, which we use as semantic knowledge base.
3. The model stores **personal interests** represented as complex semantic items which have certain properties associated with them (e.g. locations related to activities, properties of objects, ratings, etc.).
4. The profile instances are kept over **multiple sessions** and are reusable for different applications.
5. The information is acquired by analyzing natural language answers **entered directly** by the user during a dialog interaction. It is stored in an **explicit**, semantically structured graph.

In the remainder of this section, we argue for these design decisions in more detail. The focus in this chapter is on motivation, whereas Chapter 4 describes the Profile Model in full depth.

2.1.2.1 Modeling Individual Users and Using a Target Group Model for Dialog Control

Our approach to interest acquisition can be considered part of a larger personalization movement, which tries to capture individual **taste and lifestyle** preferences of users more adequately. Many authors (e.g. [Towle and Quinn 2000, Ghani and Fano 2002]) emphasize that their recommendation generation approach could significantly profit from additional, more individual information about users. This view is substantiated by our experience that adequate, individual recommendations cannot be generated on the basis of very general profiles. A profile should match the individual preferences and characteristics of a user as precisely as possible. Example 2.1 illustrates why a generalized profile cannot meet this requirement:

Example 2.1:

If the user can only choose general attribute values like “reading” or “swimming”, these do not reflect the peculiarities these activities may show for the individual user. While one user might prefer reading her favorite type of newspaper and swimming in public swimming pools, another user might preferably read science-fiction novels and like swimming in the Mediterranean, where she goes every summer with her family.

This is also the reason why we do not use a stereotype or group model to infer a profile for a new user: the profile would be too general for generating truly personalized recommendations. We use a sociological *target group model*, however, for controlling the dialog and selecting questions about related topics (cf. Section 5.3).

After investigating a variety of target group models from sociology and marketing (e.g. [Schulze 1992, Spellerberg and Berger-Schmitt 1998, Sinus-MilieusTM by Sinus Sociovision, www.sinus-sociovision.de]), we chose the model developed by Schulze [1992]. Our approach is not dependent on this particular target group model, however. Any target group model that uses a similar group description, which most of them seem to do, can be employed in LINGUINI. The following central properties of the target group model by [Schulze 1992] made it most suitable for our purposes:

- It models personal interests and dislikes of specific groups in great detail.
- Interests and dislikes are represented in a similar syntactic structure to our user input, which makes the user input easily comparable to target group model attributes.
- It is noncommercial and easily available for research purposes (as opposed to Sinus MilieusTM for example).

It might be regarded as a sociological disadvantage that this target group model consists of only five *milieus* (cf. Section 2.1.1.4), since it might not cover all characteristics of modern society. However, this circumstance does not affect our approach, which requires at least two target groups, but allows for an arbitrary number of target groups greater than two.

2.1.2.2 Explicit Profile Information

Interacting directly with the user to acquire reliable information argues for an approach that stores this information explicitly and makes it visible and modifiable for the user. The most important reasons for using an explicit user model are directly connected with the main requirements mentioned at the beginning of Section 2.1.2, especially

- the need to model **semantic relations** between items, which is best accomplished by a semantic graph structure (cf. Section 2.1.2.5)
- **sharing** profile information between applications, which is boosted by an explicit profile structure

Another reason is **transparency**. As we have outlined above, explicit profiles and the active integration of the user into the profiling process [McNee et al. 2003] allow for better transparency, which is expected to increase trust, loyalty, and cooperation of the user.

There are no objections to using this explicit profile information together with user data acquired prior to the mobile situation, for example, during previous online sessions at the desktop. Such information might, for instance, be implicitly collected by observing internet Websites the user has visited. On the other hand, additional information may also be explicit,

such as data of simple structure (e.g. age, gender) acquired by means of an online questionnaire.

2.1.2.3 Long-Term User Profiles for Different Applications

The fact that most existing adaptive systems acquire profile information only for their own purposes and often only for a single session imposes considerable effort on the user, who has to enter similar information over and over again. With the growing number of easily accessible online offers, users become less willing to take this effort. We argue that future profiling approaches therefore need to minimize this burden by enabling reusability of user information.

Our Profile Model is designed to allow multiple usage by different applications as shown in Figure 2.3. This is supported by our explicit profile representation and the use of XML as modeling language. We store the profiles on a central server (shown in the middle of Figure 2.3) where they can be accessed by different services and applications:

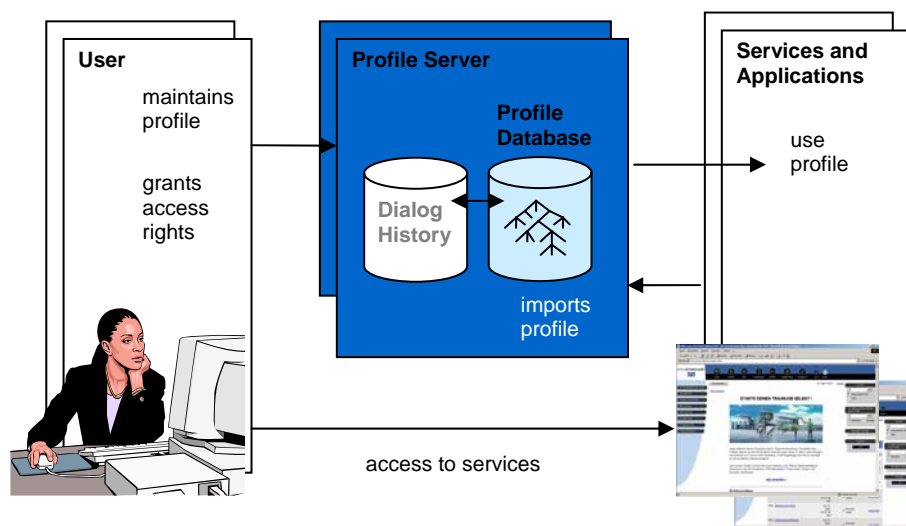


Figure 2.3: Central storage and multiple usage of user profiles

In particular, we want users to be able to see and modify their data and to grant access rights for specific topic areas or individual attributes in the profile graph (cf. Figure 2.3, left). We follow Koch [2002] by arguing that users should be the owners of their profiles, i.e. they should be able to decide which services have access to their data and should be informed when an access takes place. The functionality of importing existing profile information from other applications mentioned in [Koch and Möslein 2003] can be combined with our approach. Moreover, a dialog history is stored for each user on the central server, such that the dialog can be continued and the profile can be updated in succeeding sessions. The sharing of profile information is also supported by the chosen type of contents, personal interests, as we will see in the next section.

2.1.2.4 Personal Interests

In this thesis, we focus on personal interests for mainly two reasons:

- As mentioned above, it has often been emphasized (e.g. [Towle and Quinn 2000, Ghani and Fano 2002]) that the performance of personalized recommender systems

could be increased significantly by acquiring more individual information about users. Especially, **semantically complex information**, such as **interests**, attitudes, relations, taste, lifestyle, and the like, would be desirable for a variety of adaptive systems in order to tailor their offers and recommendations more adequately to the user's individual preferences with respect to these categories. However, there are almost no systems yet that acquire such semantically complex user information (cf. [Koch 2002]).

- Personal interests are very suitable for the **acquisition by means of an adaptive natural language dialog**. An adaptive dialog is particularly suitable for handling the semantically complex structure of interests, because the details about an interest can be acquired step by step during the interaction. The acquisition of interests from free natural language user input is not envisioned in most existing explicit profiling approaches. New methods have to be developed to approach this task.

We explain these observations in the following. First, we would like to comment on the usefulness of interest information for adaptive systems. Personal interests are not only relevant for our application scenario of event and sightseeing recommendations, but for various other types of **recommender systems**. Stolze and Ströbel [2004] discuss how the recommendation of technical products can be enhanced by not only focusing on preferable product features, but also on the user's high-level needs. Personal interests can, for instance, be used to recommend **products**, such as books, music, videos/DVDs, or **travel options** by taking into account preferred countries, leisure activities, etc. Moreover, interests can help to recommend **gifts**. Here, the user might maintain separate profiles for the interests of different persons for whom she wants to get a gift recommendation. Interest information is also beneficial for **virtual communities**. In mobile lifestyle communities [Reichwald et. al 2001, Koch et al. 2002], for example, members can find other like-minded members by browsing their interest profiles. Interest profiles may even be matched automatically by the system to suggest like-minded users. Other areas where interests play a role are **personalized lifestyle-based advertisement** (e.g. [Lekakos and Giaglis 2002]) and **personalized online news**. Finally, Websites of **dating agencies** (e.g. MatchNet, www.matchnet.de) make use of interest information, such as locations the user likes to visit, leisure activities and sports, reading habits, favorite food, or music.

The second observation was that complex interest information requires an alternative acquisition method. Personal interests are not suitable for acquisition by means of observing user actions, because the information acquired with this method is not reliable enough (cf. Section 2.2.2.1). This implies two things: First, a **more reliable method** has to be chosen. Such a method may be to have users directly enter information. Since users are involved directly, the acquired information should also be made transparent for them in form of an explicit profile. Secondly, if the aim is to acquire **highly individual interests**, the profiling method has to allow for **free user input**. In an empirical study (cf. Section 6.1), we asked users to enter their interests without imposing a specific structure and found that the interest formulations they chose show a complex semantic structure which is ideal for dialog acquisition and for asking refining questions.

2.1.2.5 Generating Profiles Dynamically on the Basis of a Lexical-Semantic Network

The Profile Model developed for this thesis represents user interests in the form of an explicit semantically structured graph. It is flexible enough to allow for an individual graph structure for each user according to her specific interests. A user profile created on the basis of the Profile Model is a subgraph of the lexical-semantic network GermaNet (cf. Section 3.5.1), ex-

tended with complex semantic nodes representing user interests. The model allows for a flexible extension with new interests and details about existing interests during the dialog. Both preferences and dislikes can be represented, and personal ratings mentioned by the user can be added to the interests.

In this section we only give reasons for this Profile Model, whereas in Section 4.4 we explain its structure in full detail. Complex semantic nodes are required for representing interests, because interests – as opposed to basic customer data like name, address, age, etc. – cannot be represented by simple character strings, but require complex and extensible data structures. In our Profile Model, each complex semantic node represents one user interest and consists of attribute-value pairs that further characterize this interest. Using attribute-value pairs is common in explicit profiles and has certain advantages. [Ghani and Fano 2002] mention, for instance, that on the basis of explicit attribute values in the profile explanations can be given to the user as to why a product has been recommended. Figure 2.4 shows a simplified sample interest node in our approach (note that further attributes are possible depending on the input):

User input:	regelmäßig Tennis spielen mit meiner Schwester (regularly playing tennis with my sister)
Attributes:	object = Tennis (tennis) activity = spielen (playing) participants = mit meiner Schwester (with my sister) frequency = regelmäßig (regularly)

Figure 2.4: Simplified sample interest node with attribute-value pairs

The assumption that truly individualized profiles are required in order to generate highly adequate recommendations (also cf. [Towle and Quinn 2000]) has led us to dynamically create an individual profile graph for each user. As a consequence, the structure of the profile instances can look significantly different and the topic areas covered are determined to a major degree by the individual user.

A main reason for using **GermaNet** as basis for our model lies in the fact that we had to find a possibility to insert new, not predefined information provided during the dialog correctly into the profile structure. In LINGUINI, GermaNet serves as a semantic knowledge base which helps to handle free user input. Another important reason was that the relations modeled in GermaNet allow us to make inferences about the semantic relatedness of interests. So, we can ask questions about closely related interests during the dialog first. If the user enters, for example, “Geige (violin)”, we can infer that this is a stringed instrument and may ask the next question about stringed instruments in general or sibling nodes of “Geige (violin)” such as bass or cello. In Section 5.4, we will see how this question selection process works in detail.

It can be observed that only few explicit profiling approaches employ a graph or ontological structure at all, but that this topic seems to slowly gain more importance (e.g. [Denaux et al. 2005]). Most currently existing profiling approaches in this area are implicit ones. In implicit profiling, ontology-based user profiles can, for example, be derived from browsing documents in the Web (e.g. [Pretschner and Gauch 1999, Magnini and Strapparava 2001]). An important topic with respect to **explicit** ontology-based profiling is the exchangeability of profile information. Mehta et al. [2005], for instance, propose an ontology-based user context model which supports the exchange of profiles between applications.

2.1.3 Applications Using Profiles: Personalized Recommender Systems

A special type of adaptive system which employs user information are personalized recommender systems. Since we focus on recommender systems in our sample application scenario, we will explain their underlying principles and methods in more detail here. *Personalized recommender systems* generate recommendations on the basis of user profiles, as opposed to *non-personalized recommender systems*, which do not operate on user information but only provide standard recommendations. We will first give an introduction to personalized recommender systems and will then concentrate on recommendation of tourist information, because this is the sample scenario in which we want to apply our profiling system. For an extensive survey about E-Commerce recommender systems cf. Schafer et al. [2000].

2.1.3.1 Introduction to Personalized Recommender Systems

What is a Personalized Recommender System?

The aim of *personalized recommender systems (PRS)* is to recommend information items or products that are of particular interest for an individual user. They are intended to respond directly to consumers by giving them advice and are modeled after the principle of “word of mouth” [Schafer et al. 2000]. By only recommending products which are relevant for the user, they aim at achieving a win-win situation: satisfaction on the user’s side is supposed to correlate with a stronger loyalty and a higher product purchase rate. Personalized recommendations are generated on the basis of implicit or explicit information collected about users. Therefore, the topic of acquiring user information is central for each PRS. Filtering methods are then used to compare user profiles with each other (e.g. for recommending products which another user with a similar profile has bought) and with a product model or database in which product information or information items are stored. The aim of this filtering process is to select an item which is of particular interest to the user on the basis of her current profile.

Different Types of PRS

[Schafer et al. 2000] distinguish two types of PRS: First, PRS which have an *offline phase* during which they learn about user preferences and an *online phase* in which they apply the learned information for generating recommendations. Secondly, there are PRS which *learn user preferences while making recommendations* in real time. The majority of existing systems today belongs to the second type. PRS also differ according to the products for which the recommendations are generated. User interfaces and functionality of PRS may, moreover, be influenced by the specific business goals of the vendor.

In the following, we describe application areas in which PRS are typically employed and present examples for items and products which can be recommended. PRS can, for instance, support users during the **online configuration of products**. Here, manufacturers help their customers find the most suitable product variant by means of Web-based configurator tools [Sabin and Weigel 1998, Ardissono et al. 2001]. Such configurator tools are used, for instance, by vendors of computer hardware (e.g. Dell), automobiles (e.g. Audi), or even by apparel vendors (e.g. Adidas). The configuration of complex products, where users can choose from a large set of properties and features, can be supported by integrating a recommender system which preselects certain variants. However, for duplicate parts a configurator normally only provides a simple choice of variants. In the area of mass customization, however, where users can specify more product properties (e.g. of a personalized domestic robot), the configuration process is also more complex and may easily overstrain the user. Automatically generated personalized recommendations for products and components are therefore particularly

useful in the area of mass customization and can help to reduce the experienced complexity [Stegmann et al. 2003, Leckner et al. 2004].

In E-Commerce, PRS are also popular by vendors of **books, movies, and music** (e.g. Amazon, www.amazon.com). The recommender system on Amazon's Website, for instance, recommends books on the basis of other preferred books of a customer [Sarwar et al. 2000]. COGITO, a research prototype for book recommendations, has been tested with the vendor BOL Medien (www.bol.de, subsidiary of Bertelsman AG) [Abbattista et al. 2002b]. Further E-Commerce applications, which integrate PRS to support **online auctioneers** or provide recommendations for **drugstore products** for example, are described by Schafer et al. [2000].

There are also **non-commercial applications**, such as movie databases, where users can get personalized recommendations after rating a number of movies they have seen (e.g. MovieLens, movielens.umn.edu). MovieLens has been developed on the technological basis of the GroupLens project [Resnick et al. 1994], one of the largest research projects about recommender systems.

The application of PRS in **mobile scenarios** is increasingly being investigated. For mobile applications, context awareness is of particular interest, which is supposed to lead to a selection of information that is relevant for the user's physical location [Ardissono et al. 2003]. Another aspect is that the acquisition of user profiles, needed for generating recommendations, can take place in different locations using multiple devices. Autonomy (www.autonomy.com), for instance, has developed the Commerce Application Builder™ for this purpose. This technology supports the user in generating a profile via PDA, for example, which can then be used to recommend Internet or news content via Email or SMS. The advantages of mobile recommender systems have also been investigated by Miller et al. [2003] in an extensive field study conducted for the MovieLens project. For this purpose MovieLens was accessible on a cell phone browser, an AvantGo¹ channel, a wireless PDA, and a voice-only phone interface. Billsus et al. [2002] describe requirements for mobile adaptive systems and recommender systems. They argue that a literal translation of adaptive services to the mobile scenario is not possible. Not only the presentation of data has to be adapted to smaller screens, but also specific implicit and explicit profiling methods have to be employed in a combined fashion [Billsus et al. 2002].

How Are Recommendations Generated?

Recommendations are generated by means of *filtering methods* that select relevant information from a product database, a document repository, the Internet, etc. While some systems only make simple database queries, others use more sophisticated filtering techniques, such as nearest neighbor algorithms or Bayesian analysis [Schafer et al. 2000].

The main techniques used for generating recommendations are content-based and collaborative filtering [Balbanovic and Shoham 1997]. *Content-based filtering* operates on the contents of products or information items. It typically extracts attribute values from the products or information items and compares them with attribute values in the user profile. Content information can, for instance, be extracted or accessed by analyzing information items (e.g. documents) or by means of meta data tags, which are entered for each product in a database. Meta data tags can, for instance, be categories for books, such as science-fiction, thriller, etc.

In contrast to content-based filtering, *collaborative filtering* tries to find similar users and then recommends objects liked by one of these like-minded users [Good et al. 1999]. The collaborative filter analyzes profiles of other users. The more similar a customer, the more relevant are her previously bought products or liked items. The main assumption of this filter is

¹ AvantGo is a service for the synchronization of Web pages, with which information can be accessed with a PDA, even though the user is disconnected from the network [Miller et al. 2003].

that like-minded customers will tend to buy similar products or like similar items (cf. [Breese et al. 1998]).

Both methods have advantages and disadvantages. For content-based filtering, the contents of objects have to be analyzed manually or automatically and stored in a suitable meta-data representation. User profile information has to be made available in a way that allows for comparison with this meta information. Changes in the product or items database can require a reiteration of this process and in some cases also an adaptation of the attributes used. The main shortcoming of collaborative filtering, on the other hand, is that it is subject to the Cold Start Problem as mentioned earlier in this chapter, i.e. it cannot be applied if there are only a few user profiles available. Furthermore, each specialized filtering method explores the user's profile only partially. A collaborative filter, for instance, makes use of inter-customer relations, but ignores the customer's preferences directly leading to specific product properties. Therefore, employing only a single filtering method will leave important information about the customer unused. As a consequence, recent approaches suggest a **combination of filtering methods** (e.g. [Billsus and Pazzani 1999, Good et al. 1999, Burke 2002, Melville et al. 2002, Schein et al. 2002]). Renneberg and Borghoff [2003] and Stegmann et al. [2003], for instance, suggest an integrated filtering approach that tries to solve some of the problems mentioned above. They describe a filtering pipeline which combines filters with different "responsibilities", i.e. filters are specialized on certain parts of the product model.

In Section 2.1.2.4 we have discussed adaptive systems for which **personal interests** of the user are of particular relevance. PRS definitely belong to this category. In PRS, interest information can, for instance, be matched either collaboratively against the interests stored in other user profiles or with a content-based method against meta information stored about products or information items. The exploitation of interest information is especially interesting for generating "**cross category recommendations**" (cf. [Ghani and Fano 2002]). In the approach of Ghani and Fano [2002], for example, explicit profile information is used to recommend both apparel and furniture. We would like to illustrate the idea of generating recommendations across categories considering the example "gift recommendation". Imagine a user is seeking a gift for a person who particularly likes reading short stories in bed. Then this profile information cannot only be used for recommending **books** the person might like, but also for **domestic objects** which are related to the entire activity, e.g. a cushion, reading lamp, or even a hammock. The interests acquired by means of LINGUINI are especially useful for this type of cross category recommendation, because they may span a wide range of topics and take into account a variety of details for a specific interest, such as location, rating, participants, properties, etc.

2.1.3.2 Recommending Tourist Information

After this overview of PRS, we will now take a closer look at recommending tourist information, being mostly related to our application scenario. The central idea of a tourist PRS is to help users select from the vast choice of attractions (e.g. leisure activities, events, or places to visit) offered for a specific town or area. For this purpose, the PRS may either acquire personal information prior to the trip (e.g. via Internet), while the user travels (e.g. in a car with Internet access), or when the user has arrived at her travel goal and walks around with her mobile device. In a second step, the PRS can use this profile information to provide personalized recommendations. Some of the existing systems in this area (e.g. INTRIGUE [Ardissono et al. 2003], PolyLens [O'Connor et al. 2001], AVANTI [Fink et al. 1996]) allow for not only supporting individual users, but also user groups, since they argue that users seldom travel alone [Ardissono et al. 2003].

In Section 2.1.1.2, we have discussed requirements for mobile information access. Some tourist PRS, such as GUIDE described by Cheverest et al. [2000], require specialized hand-

held devices and hardware which is located in the town to be visited. GUIDE uses a cell-based wireless infrastructure to broadcast location and dynamic information between server and hand-held device. Other systems, such as INTRIGUE, presented by [Ardissono et al. 2003], are more flexible. INTRIGUE is based on standard wireless connections and the tourists can use their own mobile phones.

Existing tourist PRS also differ with respect to their interaction and profiling approaches. We will present some applications here and discuss the differences to our profiling system. Fink et al. [1996] describe AVANTI, a system which aims at tailoring the content and presentation of Webpages to users with different needs. AVANTI provides personalized information about a metropolitan area (e.g. public services, transportation, buildings) for tourists, citizens, travel agency clerks, people with specific handicaps, etc. AVANTI is supposed to be accessible in various ways, for instance, from desktop computers, public information booths, or hand-held devices. The system explicitly models user information such as background knowledge, aims, interests, or abilities. The user is initially interviewed and assigned to a stereotype. Further user information can be implied indirectly from user input entered during a dialog interaction (e.g. if the user requests an explanation for a technical term, she is probably not familiar with it [Fink et al. 1996]). As opposed to the AVANTI approach, the recommendations we have in mind are much more personalized and require highly individual user information, which exceeds the possibilities of a stereotype model.

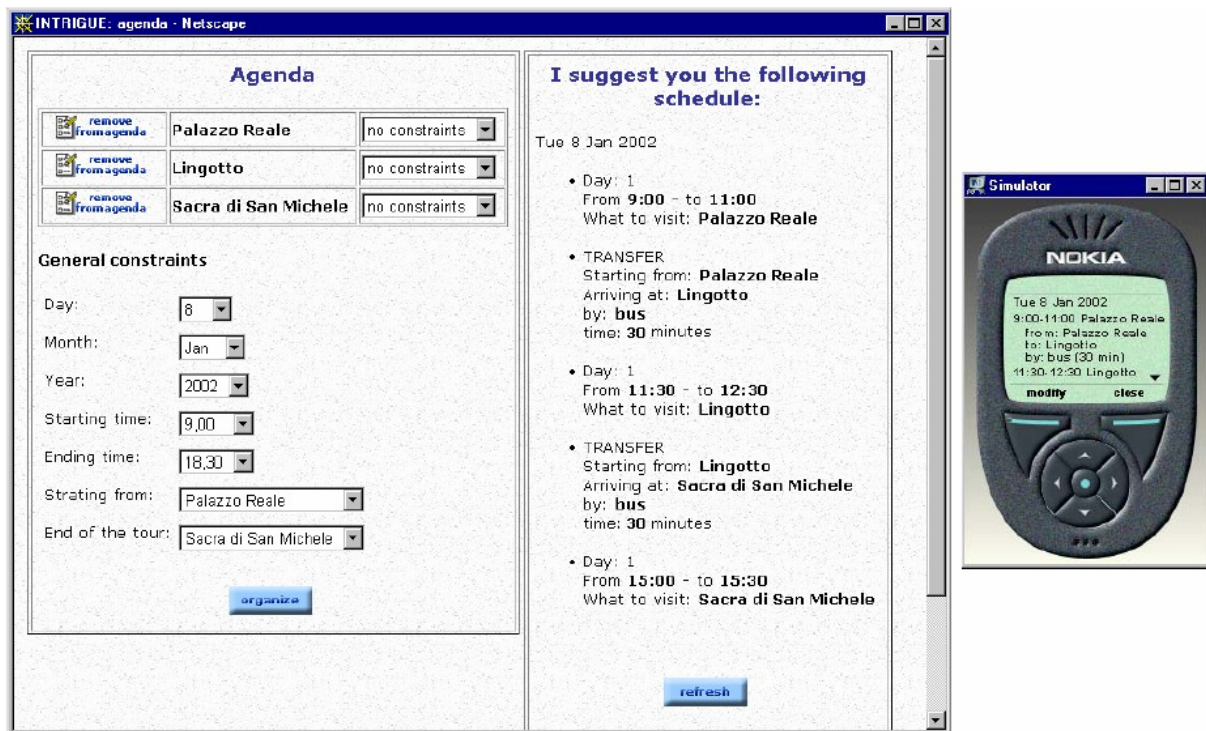


Figure 2.5: Specification of constraints and tour presentation in INTRIGUE (source: [Ardissono et al. 2003])

Another approach is INTRIGUE (“Interactive Tourist Information Guide”), which provides personalized information about the area around the city of Torino in Italy. The PRS can be accessed from desktop or handset devices and provides ubiquitous assistance for organizing sightseeing tours. The approach of Ardissono et al. [2003] differs from our approach in several respects. First, it adapts recommendations to user groups, whereas in our approach the

individual user is addressed. Secondly, the structured group model employed by Ardissono et al. [2003] is application dependent in that there is a one-to-one mapping between user preferences and properties used for describing tourist attractions in the INTRIGUE database (cf. Figure 2.5 above). As opposed to this, our user profiles have an application-independent semantic representation and can hence be used for different tourist information databases. Finally, the INTRIGUE approach focuses on visualization of tourist information on mobile devices, whereas our approach concentrates on verbalization, i.e. the interaction with the user via speech dialog.

GETESS (“German Text Exploitation and Search System”) is an agent-based approach that uses natural language processing techniques for collecting user preferences and for presenting information in a user-friendly manner [Staab et al. 1999]. The system provides information filtering methods which Staab et al. [1999] exemplify in the domain of sightseeing and event recommendations. The system is supposed to take into account tourist-relevant information from different sources found on the Internet. In GETESS, users interact with the system by means of a natural language dialog in which they request certain types of information (e.g. “Which cultural events take place in Mecklenburg-Vorpommern?”), for which the system provides recommendations (e.g. “Rostock, concert series of the Hochschule für Musik und Theater”). Although the system is supposed to be applicable to “many realistic scenarios” [Staab et al. 1999], the profiling and language processing process depends on domain knowledge which needs to be adapted for each new domain. In our approach, the language processing component is tailored to the acquisition of interest information, and hence could not be used for conducting dialogs about other topics, but it is not dependent on a specific application domain (e.g. tourist recommendations).

2.2 Profile Acquisition Methods

In the first part of this chapter, we have discussed issues in user modeling, types of user models, and applications using profile information. In the second part, we now address the topic of acquiring this information. We will start with outlining challenges for acquiring profile information in Section 2.2.1. In Section 2.2.2, implicit and explicit profiling methods are introduced. Since this thesis suggests a natural language profiling dialog, Section 2.2.3 is dedicated to adaptive systems and existing profiling approaches that involve natural language. Section 2.2.4 concludes this chapter by giving reasons for our acquisition approach on the background of existing methods.

2.2.1 Challenges for Acquiring Profile Information

We found several challenges for acquiring profile information during our investigations, which we summarize in this section. First, the result and quality of a personalization service seems to depend directly on the **profile quality**. A product recommendation, for example, will be as adequate for an individual user as is the information stored in her user profile (and the inference mechanisms applied). If little is known about the user or if the information is very unreliable, then highly-suitable recommendations are an unachievable goal. As a consequence, the primary goal has to be the acquisition of profiles of a high quality, i.e. which contain sufficient and reliable information that represents the user preferences as adequately as possible.

The second observation is that a certain **compromise** is necessary between the amount and quality of information collected and the effort it takes for the user to enter this information. Many Internet vendors’ personalization services suffer from the poor quality of user information acquired by just observing the user’s behavior. On the other hand, it is risky for

them to ask users directly for (more reliable) information, because they might not want to take the effort of filling in large questionnaires and might switch to another vendor.

Another observation concerns the **degree of dependency** of the profiling method on the adaptive system. Here, two types of systems can be distinguished. First, systems which communicate with the user *while offering adaptive services* (e.g. [Abbattista et al. 2002b, Thompson et al. 2004]). Their user model is in general *application-dependent* and the personalization methods are adjusted to this particular model and its data structure. Some of these systems immediately improve their services, e.g. the recommendations given, by means of user feedback. An example is the Needs-Based Recommender described by Stolze and Ströbel [2004], which offers three profiling phases while recommending items in between. During the profiling phases, users have the chance to refine their product preferences. Further examples are the Automated Travel Assistant [Linden et al. 1997], or the system presented by Ardisson [1996], which uses clarification dialogs to update user information during a natural language dialog that aims at supporting students by organizing their studies.

The second type of system, to which the approach of this thesis also belongs, *keeps profiling and personalization methods separate* or constitutes an *independent profiling system* (e.g. GUMS [Finin 1989], BGP-MS [Kobsa 2001], OntobUM [Razmerita et al. 2003]). The main reason for doing so is to keep the profiling part *application-independent* and hence reusable. Following this approach, the profiling phase may take place before the actual personalization service. If the results of the service do not yet meet the customer's expectations, another profiling phase can be added. Unfortunately, it is not possible to profit directly from user feedback provided during the recommendation process with this approach. Hence, a combination of both approaches might be a promising solution.

Application-independent systems are strikingly less common than the application-dependent ones. We believe that the trend goes towards exchangeable profiles (cf. [Kobsa 2001, Koch 2002]), however, which are owned and managed by the user and can be reused for different types of applications. This task not only requires **more flexible user models**, but is also challenging because it needs to take into account privacy issues and make profiles transparent in order to increase the users' trust and cooperation.

Another challenge is that **new or not yet fully explored application situations** may require alternative profiling methods. We tried to illustrate this previously by means of the vehicle scenario, where profiling questionnaires cannot be employed. In the succeeding sections, we will see how existing approaches cope with respect to these challenges.

2.2.2 Implicit and Explicit Profile Acquisition

In Section 2.1, we discussed characteristics of implicit and explicit user models. The differences between these two types of models become even clearer when investigating the way the information is acquired. The techniques for acquiring profile information require the engagement of the user to different degrees. For adaptive systems, it is common to distinguish implicit and explicit profiling methods [Brusilovsky and Maybury 2002]. For *implicit profiling*, the system monitors the user in her browsing or shopping behavior and determines her interests by using information clustering techniques. For *explicit profiling*, the user is asked to enter personal information and preferences directly. The two types of methods are described in detail in the following.

2.2.2.1 Methods for Implicit Profiling

Most adaptive systems in E-Commerce use implicitly collected user information, i.e. they observe the user's behavior and transactions [Kobsa 2001]. This is due to the risk of overstraining customers when asking them to enter personal information directly. Implicit profil-

ing is an unobtrusive approach [Thompson et al. 2004] and this also explains why a lot of research has been done on implicit profiling methods, whereas there are comparatively fewer approaches to explicit profiling.

For implicit profiling, online shops usually record transactions in a database. This can be done both online and offline. Large offline retail shops have introduced membership card programs to identify customers during their purchase transactions and to keep an identified log on their transactions. In addition to information about transactions, online shops store information about the browsing behavior of customers. Page visits can be tracked and the time a customer spends on a particular page can be stored. A main problem with tracking the browsing behavior is the identification of the customer. The information about the IP address of the requesting client is often insufficient for identification due to the use of dynamic IP addresses. Therefore, current Websites often try to solve this problem by setting local browser cookies.

Machine learning techniques are then applied to draw inferences from the collected data. Personalization often directly takes place on this basis without storing any of the user information that has led to the assumptions. There is a variety of machine learning approaches for user profiling. An overview can be found in [Abbattista et al. 2002a].

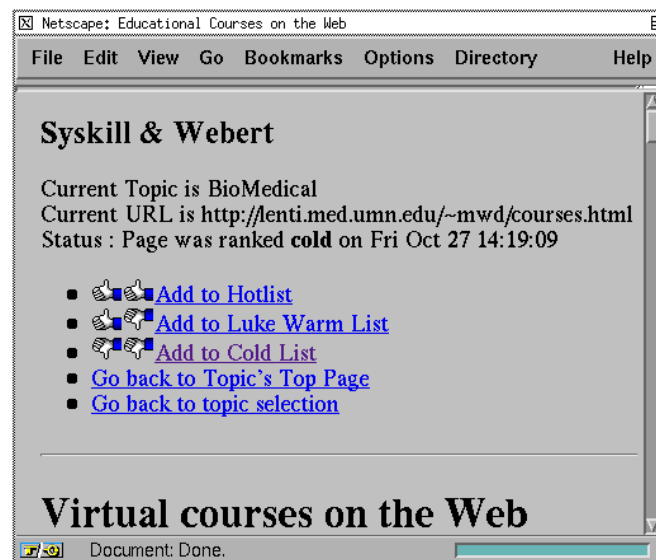


Figure 2.6: Syskill & Webert interface for rating pages (source: [Pazzani et al. 1996])

Implicit user information can be extracted from various sources. Some systems (e.g. Commerce Application Builder™ by Autonomy, www.autonomy.com) observe the user's **browsing behavior** on the Web or on her local computer and make recommendations on the basis of documents she has looked at. It is also possible to **learn from ratings** given by the user, which may lead to more reliable information than that gained by observation. The Syskill & Webert agent [Pazzani et al. 1996] learns from Web pages the user has visited and rated (cf. Figure 2.6 above). On this basis, it suggests links the user might find interesting.

Billsus and Pazzani [1999] present the News Dude system, which reads news stories retrieved from various Internet news channels to the user. The system uses a multi-strategy learning approach which learns both short-term interests (by means of the nearest neighbor algorithm) and long-term interests (with a naïve Bayesian classifier) of the user. The information is learned from a set of news stories the user has rated by giving feedback to the system. Since for a learning algorithm the selection of items to be rated is crucial, Rashid et al. [2002] discuss different strategies for selecting and presenting items to the user. They show, for instance, that this process can be optimized by also taking into account the user's experience.

Other approaches evaluate the user's **behavior when interacting with the application**. This method is often used to infer the user's goals and beliefs during a dialog interaction with the system (e.g. [Pohl et al. 1995]). Although most implicit methods do not lead to an explicit information representation, there are some approaches, however, that **learn explicit user profiles** (e.g. [Ghani and Fano 2002, Schwab and Kobsa 2002]).

The main advantage of implicit profiling is that it imposes little or no effort on the user. Although this feature is indeed very convincing, many drawbacks of implicit profiling have also been pointed out in literature. Implicit profiles may, for instance, suffer from a **lack of accuracy and reliability** [Brusilovsky 1996]. Since implicit user information alone seems insufficient for generating highly adequate recommendations, many authors request at least a combination with explicit information (e.g. [Ardissono 1996, Towle and Quinn 2000]). Another problem is that **not all observed user actions have the same relevance** for learning user preferences. Users occasionally only want to try out something on the system. Moreover, new users may behave in an inexperienced manner. These situations may lead to wrong facts learned by the system [Mertens and Höhl 1999]. Another problem is **missing transparency**. Implicitly collected information often cannot be made transparent to the user for corrections. Erroneous information will hence remain as part of the profile in case no updating method is able to correct it automatically. Moreover, erroneous system actions based on an invisible user profile are often irreproducible for the user, which may increase mistrust. It is therefore recommended to visualize the profile for the user [Thompson et al. 2004]. The importance and advantages of transparency in user profiling are discussed in detail in [Bauer 2004]. Another observation is that implicit information in most cases is **product-related** and can only be used by the application which collects it. An exchange with other applications is not possible in general. The fact that most implicitly collected information is not represented explicitly also prevents it from being stored on a central server for sharing and exchange [Kobsa 2001]. Finally, as pointed out in [Brusilovsky 1996], **not all types of user-related information can be acquired implicitly**. Information which cannot be acquired implicitly is, for instance, name, address, gender, age, and other basic personal information as well as the user's background (e.g. profession, experience of work in related areas to the application, or point of view). Therefore, it seems important to investigate more intensely methods for explicit profiling.

2.2.2.2 Methods for Explicit Profiling

We will now discuss explicit user profiling, where users are directly asked by the system to enter their preferences. Information obtained with this method is typically stored in an explicit user profile.

The most widely used technique for explicit profiling is **online questionnaires** or **forms**. Here, static and adaptive questionnaires can be distinguished. *Static questionnaires* look alike for each user. They offer a fixed set of question slots and are particularly suitable for acquiring basic personal data of simple structure (e.g. name, address, age, bank account, etc). Users are typically asked to fill in a questionnaire, if they want to register for a specific service or application, or if they request a recommendation for the first time. Many E-Commerce platforms acquire explicit information with this method. MyYahoo (www.yahoo.com) and Amazon (www.amazon.com), for instance, create structured explicit profiles with this method, which are visible for the user.

A drawback of static questionnaires is that the presented questions and answer possibilities do not fit each user alike and that filling in a large form can be tedious. *Adaptive questionnaires* try to address this problem by reducing the pool of required questions [O'Nuallain and Redfern 2005]. EDUFORM, for instance, is an adaptive questionnaire which selects questions presented to students adaptively. EDUFORM clusters previously collected profiles into

groups. On the basis of the answers a student enters, the similarity to one of the groups is calculated. The group profile is then used to optimize the order in which the questions are presented. The student may quit, once sufficient certainty about her profile has been achieved [Miettinen et al 2002]. This method allows determining the student's profile with a minimal number of questions. In the approach of this thesis, we also use a group model to select meaningful next questions. For creating the group model we do not need previously collected profiles, however. Moreover, we do not stop the profiling process as soon as a user can be assigned to one of the groups. Our aim is to collect as much individual information as the user is willing to give instead of adopting a rather general group model as user profile.

Adaptive questionnaires are related to the field of *computerized adaptive testing*. Novell, for instance, explains on its Website (www.novell.com) how it uses adaptive testing for finding out the expertise of participants of technical trainings. The idea is that the computerized adaptive test tailors itself to the abilities and the level of knowledge of the test person. This is done by calculations made on the basis of the answer given to an initial question. Succeeding questions are selected according to the level of difficulty that matches the test person's calculated ability. After each response the ability is recalculated and the next question is selected adaptively. The test ends, if the estimated ability has passed a certain confidence threshold.

Another very common way to acquire explicit information is **rating items** (e.g. [Pazzani et al. 1996, Billsus and Pazzani 1999, Rashid et al. 2002, McNee et al. 2003]). Here, users label, for example, products they have bought, information items that have been recommended, movies in a movie database (cf. MovieLens, movielens.umn.edu) with a value which indicates their degree of liking or disliking. Typically, the user selects a rating from a range of numeric values, but some systems offer sets of specific rating categories [Thompson et al. 2004]. Items to rate are often listed in the form of questionnaires. Another possibility is to request a rating for each item presented in a recommendation process or during a dialog interaction. The collaborative filtering technique of personalized recommender systems is often based on ratings. It generates recommendations for a user on the basis of other profiles which are similar with respect to their ratings.

User preferences can also be acquired by conducting a **natural language dialog** with the user. There are typically two favorable points of time for conducting a profiling dialog. It may either take place *while* the user is navigating on a Website, items are recommended, or services are adapted, or it may be conducted *before* the actual recommendation or adaptation. Most dialogs center on product preferences or try to elicit customer goals and intentions with respect to a specific application. The dialog method is particularly suitable for acquiring parts of a more complex information unit step by step (e.g. details about a specific user interest or the user's goals in a travel scheduling task). Since natural language dialog allows for more complex user input than single words or selection lists, the main technical difficulty lies in the interpretation of this information. We discuss this profiling method and existing applications in full detail in Section 2.2.3.

Explicit profiling is always connected with a certain amount of effort on the user's side. A solution which is therefore often applied in explicit approaches – but also in implicit ones – is the use of **stereotypes** or **target group models** (cf. Section 2.1.1.4). The idea is to ask the user for as little information as is necessary in order to assign her to a most similar user group or stereotype and to infer the rest of the information from the target group or stereotype model. The GRUNDY system [Rich 1979], for instance, was one of the first stereotype systems. It asks the user to give a list of words that characterize her and, on this basis, assigns her to a stereotype.

A more recent approach, which uses a model based on lifestyle and purchase behavior of the U.S. population and which is related to our work, is Lifestyle Finder [Krulwich 1997]. Lifestyle Finder explicitly collects information about users and groups them into “demographic clusters”. The relation to our approach is that Lifestyle Finder also asks profiling

questions and uses a model of demographic clusters, which is comparable to a target group model. There are two major differences, however. First, Lifestyle Finder assigns a user to the most likely cluster and adopts the cluster profile for this user, but it does not acquire and store individual information for each user. We create personalized profiles with individual interests, attributes, and values. The second main difference is that Lifestyle Finder only offers a small predefined set of five to six answers, which are shown as graphic buttons to the users and from which they have to select (cf. Figure 2.7). In our approach, we use natural language processing techniques to also analyze free, not pre-defined answers given by the user. The author substantiates our approach in that he states that his method suffers from a loss of accuracy in recommendations and emphasizes that more detailed profiles would be desirable [Krulwich 1997].

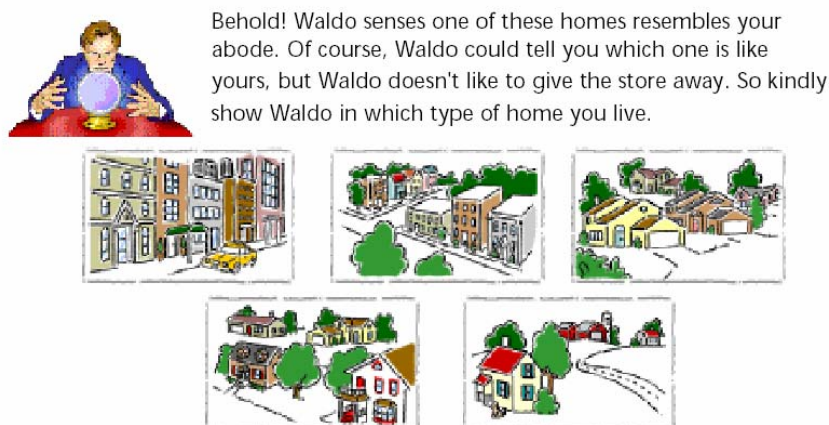


Figure 2.7: Lifestyle Finder – alias Waldo the Web Wizard – asking the user a question (source: [Krulwich 1997])

The main advantage of explicit profile information is its reliability as compared to implicit information, which is often too vague or inadequately characterizes the user on the basis of misinterpreted user actions. Further advantages lie, as mentioned already, in the possible transparency for the user and in the exchangeability and reusability by different applications.

Although the benefit of explicit information for recommender systems is obvious, very few approaches actually use it. This is due to some severe disadvantages. The most obvious problem is that explicit profiling **places the burden on the users** by soliciting preference information directly and hence bears the risk that the users might switch to another vendor, where this is not necessary. The users may be put off by completing long questionnaires before they can even begin to enjoy a given service [Thompson et al. 2004]. It is therefore important to smoothly integrate any explicit profile acquisition into the user-system interaction [Kobsa et al. 2001]. This is particularly important for approaches using ratings. Recommender systems usually need **large amounts of ratings** in order to provide acceptable results and even before the user can get the first recommendation at all.

The **stereotype** or **target group model** approach definitely helps to reduce user effort, but its main disadvantage is the **lack of individual information** about a specific user, which cannot be inferred from the model. Inferences made on this basis (e.g. product recommendations) will always be very coarse, since in most existing approaches there are only a few (often less than ten) groups or stereotypes in which all customers have to fit. For this reason, stereotype information should best be combined with further user information acquired interactively [Ardissono et al. 1999].

Static questionnaires also show certain drawbacks. The lack of personalization makes filling in large forms a boring task, which may reduce the user's willingness to reveal per-

sonal data. This problem can be addressed by making acquisition techniques adaptive to individual users as is done by adaptive questionnaires or in adaptive natural language dialogs. Moreover, this acquisition technique is not equally suitable for all users alike. Schmitt and Bergmann [2001] state that static questionnaires are best suited for expert users who understand all the slot meanings in the form, but are not necessarily ideal for a novice. Finally, static questionnaires are less suitable for acquiring complex information such as attitudes, complex goals, or interests. For these, a more sophisticated acquisition technique (e.g. a dialog interaction) is needed.

A **natural language dialog**, during which information is acquired, may solve some of these problems, but it can be time-consuming. Moreover, the dialog strategy has to be able to solve communication problems such as ambiguity in the user input (cf. Section 3.6.1) and has to have reliable fallback strategies in case the system cannot “understand” the user. These are technically very difficult problems, which existing dialog systems cannot solve satisfactorily yet. Expert users therefore often prefer (short) questionnaires or implicit methods which run in the background, whereas inexperienced users often appreciate being guided and being able to use the most natural way to communicate: human language. It has to be investigated carefully in which situations a dialog interaction is preferable to alternative methods. Such a situation is given, for instance, in the vehicle and pedestrian scenario sketched in Chapter 1, where speech input and output are necessary.

A problem for both questionnaires and dialog interaction is that users are often not completely **aware of their own capabilities** with respect to the application [Kobsa et al. 2001]. They often **cannot articulate their preferences** clearly, if they are unfamiliar with the domain or application [Thompson et al. 2004]. If the questions asked are too general, the user may feel unable to overcome this barrier, which may in turn cause the termination of the interaction. A solution here is to provide examples about possible or expected user input and to adapt the questions to the knowledge of the user. Some systems present controlled queries, for instance, which aim at an objective assessment of the user. An example for such a system is SATELIT [Akoulchina and Ganascia 1997], which asks questions about concepts in a classification domain and evaluates the responses to construct a model of the user’s domain expertise.

Finally, an explicit profiling system has to handle **privacy concerns**. Directly asking the user for information increases the awareness that the system is actually using and storing personal information. The system may be perceived as intruding much more into the private sphere as is the case for implicit acquisition, which takes place in the background. Since the legal situation is such that the user must be informed about any data acquisition and storage, any acquisition method has to address the problem of how to increase trust and cooperation on the side of the user. A possible approach for increasing trust is to realize systems, where not the Internet vendor but the user herself is the owner of profile information, and where access control is completely up to the user.

A major trend in adaptive systems is to **combine implicit and explicit profiling methods** to overcome their individual drawbacks. In the BGP-MS system by Kobsa and Pohl [1995], for instance, a questionnaire approach is combined both with a stereotype model and observed user actions with respect to the application. Ardissono et al. [1999] suggest combining an implicit stereotype approach, which provides initial information when little is known about the user, with an explicit method to update the user model and collect more specific information about the user later on. An ontology-based explicit and implicit user model, which can be used by different knowledge management systems, is described by Razmerita et al. [2003]. Kuflik et al. [2005] employ a user model which is based on explicit and implicit user feedback for a museum visitor’s guide system. The reported results show that hybrid approaches in fact often outperform single-method approaches.

2.2.3 Profiling Approaches and Adaptive Systems Involving Natural Language

A special interaction method is to conduct a natural language dialog with the user. This method can be used both for offering information to users and for collecting information about them. This section gives an overview of existing approaches combining natural language interaction and user profiling, because this combination is also part of our solution.

The approaches presented differ considerably with respect to the amount of linguistic knowledge integrated and the type and depth of natural language processing technologies involved. They also use natural language for different purposes. Therefore, we will first describe profiling chat bots, which belong to the rather simplistic approaches, and will then see applications involving high-end natural language processing techniques. In the latter category, we distinguish natural language dialog systems which employ user models to *improve the dialog interaction itself* and adaptive systems that use natural language dialog merely as a tool *for collecting user information or offering adaptive services*.

2.2.3.1 User Profiling Chat Bots

Introduction to Chat Bots

Chat robots (short: *chat bots*) are software programs which act as chat users, i.e. they communicate with other chatters (humans or robots) using natural language. Many of today's chat bots have been created for the Internet Relay Chat (IRC) originally, in order to defend channels from unwanted intruders. For this purpose, a bot is set into a channel with the task to monitor everything that is said and to give warnings or ban users if they behave inadequately. Chat bots used for channel defense have been created for the execution of system commands in the first place. They only possess a very basic language processing component that is based on keyword spotting. *Keyword spotting* means that the bot reacts to particular words in the utterance of a user, which are part of a predefined keyword list and which trigger a specific reaction of the bot.

The earliest and probably most famous chat bot is ELIZA [Weizenbaum 1966], which imitates the behavior of a psychotherapist. Although ELIZA is based on a very simple language processing mechanism, "she" has been mistaken to be human by several test persons. The questions asked and utterances made by the system are conceived by the users as belonging to a psychological strategy. Similar to the IRC bots mentioned above, ELIZA technically uses minimal context and keyword spotting and reacts with preconstructed answer patterns, which are instantiated with keywords from the user utterance.

Since then, chat bots have not changed so much from a technological point of view. Some newer samples are further developed in that they show learning capabilities for words or dialog patterns, such as Billy (<http://leedberg.com/glsoft>) or MegaHal [Hutchens and Alder 1998]. Nowadays, chat bot are used for various purposes. They may, for example, hold tutoring sessions helping new users to learn about system features and to find more information. They are also employed on E-Commerce Websites to inform customers about new products, answer questions, or help with navigation. Mostly, they are visualized as funny or attractive animated assistants, animals, wizards, and the like. The idea is not only to inform but also to entertain the customer – chatting with a bot should be fun. Most of the chat bots however, are still very simple pattern matching systems without learning capabilities and their strength lies in simulating a dialog between humans rather than understanding what the user has said. The need has been identified to adapt the bot's utterances during the dialog more to the actual user interests [Pandzic 2001]. Kiwilogic (www.kiwilogic.com) offers the Bot Creator ToolTM, with which Internet vendors can create their own bots. There are many vendors and institutions

employing the Kiwilogic technology on their Websites: Volkswagen, IKEA UK, Schwarzkopf, for instance, and even the German Academic Exchange Service (DAAD). In order to create a chat bot, the vendor has to provide domain vocabulary, which is then integrated into the bot's pattern matching rules manually. According to Kiwilogic, it takes about 2-3 person months to create a bot for a new vendor.

The main drawback of chat bots is that their language understanding capabilities are very restricted and that they typically change the topic in case they do not understand what the user has said. Instead of using proper semantic processing, they are restricted to the detection of keywords and simple discourse patterns, such as greeting, question, confirmation or rejection. No natural language generation of utterances takes place, only output of pre-constructed utterances, which may be filled with keywords taken from the last user utterance. Finally, the creation of the pattern matching database requires a significant developmental effort. Since the patterns are hardcoded, the bot's reactions are not flexible and later changes are connected with additional manual effort.

Many current open-source chat bots for various application areas can be found at www.botspot.com or www.simonlaven.com. Pandzic [2001] gives a survey about chat bots and other virtual characters, focusing on their visualization and animation rather than on their language capabilities, however.

User Profiling Chat Bots

Chat bots can also be employed for collecting user information. During its interaction with the user, the bot can learn about the user's preferences and create a user profile which it updates whenever it notices a change. Another possibility is that the bot can ask users explicitly to answer some profiling questions or direct them to a questionnaire to fill in.

Most of the few existing profiling bots are commercial and no detailed information on their profiling technique is available. In research, the topic is not addressed very often either, probably because the capabilities of chat bots are considered too limited to accomplish the profiling task satisfactorily. Hence, most research activities in the area of user profiling focus rather on high-level dialog systems instead. In the following we discuss some profiling chat bots about which information is freely accessible.

[12:28:06] Judge 1: Hi Alice, How are you doing?
[12:28:20] Alice: Hello there. Everything is going extremely well.
[12:28:50] Judge 1: I'm glad to here it. Where do you live?
[12:29:11] Alice: Thanks for the information. My location is New York City. Where are you?
[12:29:38] Judge 1: In New York. How did you get here today?
[12:30:12] Alice: Tell me a little about your life in New York. Ah. Everything I need to know, I learned from my Instructor.
[12:30:44] Judge 1: Why don't you answer my question first?
[12:31:15] Alice: Don't you know? No I do not answer your question first. Do you? I answer a lot of silly questions.
[12:31:42] Judge 1: Do you still want to know about my life in New York?
[12:31:51] Alice: Yes, please tell me.
[12:32:12] Judge 1: I'm a grad student and don't have much free time. I study a lot.
[12:32:43] Alice: You are a grad student and do not have much free time? Do you get to meet a lot of people? What subject are you studying?
[12:33:37] Judge 1: I meet a lot of my colleagues. I study computer science. What do you do?
[12:34:18] Alice: Do you make long-term friends that way? Is computer science a hard subject?
Judge 1, I talk to people on the web. What do you do?
[12:34:26] Judge 1: Please, call me Ryan.

Figure 2.8: Cutout of a conversation with ALICE (source: the complete dialog can be found at http://loebner.net/Prizef/2004_Contest/Wallace.html)

An award-winning open-source chat bot with one of the largest vocabularies existing today is ALICE (www.alicebot.org). Special about ALICE is that it uses AIML (Artificial Intelligence Markup Language), a language that is based on XML and has been created especially for the development of chat bots. Figure 2.8 above shows a cutout of a conversation between ALICE and a jury member at the Loebner Prize contest 2004.

Ginsburg [2001] uses ALICE to conduct questionnaires in an online community, the ICC (Internet Chess Club). In this approach, the bot has two possibilities to acquire user data. First, it can acquire data implicitly by observing user activities. Secondly, it can switch to a “questionnaire mode” after the user has agreed to answer some questions, for instance about her opinion concerning certain community features or modules. According to Ginsburg [2001], cooperation by the user can be increased by making sure that the bot not only asks questions, but also contributes to community life, for instance by holding online tutorials. These contributions make the bot well-known and appreciated among users before the questionnaire takes place.

Another chat bot which is capable of user profiling is Cyb, developed by Agentscape (www.agentscape.de). Cyb is a personal assistant software, which is located on the users desktop, communicates in natural language, and is able to perform smaller tasks for the user, such as transferring money or searching for information on the Internet. Cyb can acquire user preferences explicitly during a dialog and also implicitly by observing users work with their computers. The created XML-based profile is stored locally on the users’ computers and is visible and modifiable for them at any time. In connection with CyMON (an agent-based platform for one-to-one CRM applications, cf. [Reitter et al. 2001]) Cyb can profit from a language processing approach which extends pattern matching with statistical part-of-speech tagging, stemming, parsing on the basis of a semantically oriented grammar, and a unification-based answer finding technique. In this respect, it exceeds the simplistic approach of typical chat bots.

Artificial Life (www.artificial-life.com) offers chat bots that create open-format user profiles which contain explicit information (e.g. about personal data collected during the dialog) as well as implicit information. The implicit information is obtained from categorizing users into demographic user groups on the basis of their utterances. Artificial Life promotes this functionality as a possibility to improve online marketing.

In summary, the information collected by chat bots can be used for improving applications (e.g. the functionalities of an online community), for marketing purposes, or for supporting the user in routine tasks. We have not found any recommender systems, however, which are based on profiles created by chat bots.

2.2.3.2 Improving Natural Language Dialog Interaction by means of User Models

In the previous section we have seen that chat bots are a technically rather simple approach to dialog and user profiling. We now consider more complex dialog systems which employ user models from an application point of view. The underlying technology of dialog systems is presented in detail in the natural language processing chapter in Section 3.7. In this section, we introduce dialog applications for which profile information fulfills the main purpose of improving their own dialog performance (Case 1 presented in Figure 2.9). These applications have to be distinguished from those which use dialog merely for the purpose of acquiring information (Case 2 in Figure 2.9), which is then used for some other adaptive service or application (e.g. for a recommender system). The effect of the acquired information on the dialog interaction is only secondary, as indicated by the thin arrow between user model and dialog system. Case 2 will be discussed in the next section.

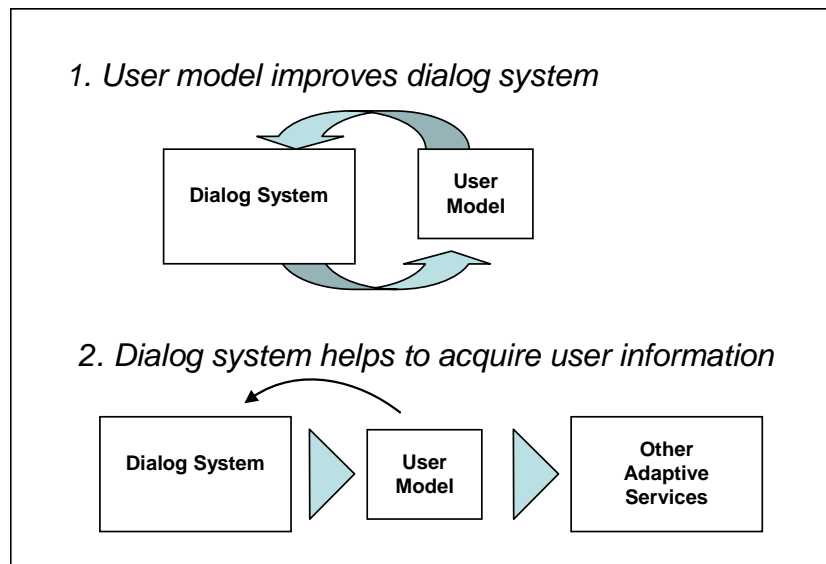


Figure 2.9: Two interaction possibilities between dialog systems and user models

During the last decades, many research dialog systems were developed as an end in itself to investigate and advance natural language processing techniques. UM was detected as a technique to adapt the dialog to the user's goals, plans, beliefs, capabilities, attitudes, etc. [Kobsa and Wahlster 1989, Zukerman and Litman 2001]. In these approaches, user models are mainly employed in order to improve the natural language interaction and not vice versa. Inferences made about user preferences can, for instance, help to generate relevant and appropriate responses, but also to analyze and understand user utterances and to extract information from them [Zukerman and Litman 2001]. Not only the interaction itself, however, but also the natural language interface can be tailored to the user's needs.

Structure and contents of the user models of these approaches strongly depend on the domain about which is spoken in the dialogs and on system tasks related to this domain. Kass and Finnin [1988] describe different information categories (i.e. goals and plans, capabilities, attitudes, and knowledge or belief) which may be contained in user profiles and how they can be used by natural language systems. Zukerman and Litman [2001] distinguish 1) user models which represent a single aspect of the user only (e.g. expertise, belief, preferences, or medical record), 2) models which comprise several such aspects, and 3) models which mainly contain deviations of the user's language from the system's language in order to be able to adapt to the user's way of speaking.

Berthold and Jameson [1999] suggested another dimension for the user model in speech dialog systems by taking into account the actual situation of the user. They call this dimension the "cognitive load" of a user, to which the dialog can be adapted by using simpler questions, etc. Taking into account the user's situation can also help the system to determine when is a good time to start a dialog interaction. However, the approach seems to be bound to speech systems, since it uses clues from the speech input of the user.

Komatani et al. [2003] investigate the generation of cooperative responses in a telephone-based collaborative dialog system. They acquire a user model implicitly by decision tree learning during the dialog. The model is used to adapt the dialog to the individual user and takes into account the following user properties: skill level with regard to the system, knowledge level within the target domain, and the degree of hastiness. The dialog strategy is changed according to the user model. If a user is categorized as having a low skill level, for instance, the system controls the dialog and prompts items in order. Otherwise, a dialog which allows more user initiative is carried out. The hastiness factor determines whether the system

asks for confirmation about its assumptions or whether these are omitted because the user probably has no time to answer them. Another application in which the user-system dialog is dynamically adapted to the amount of knowledge the user has within the specific domain is discussed by Ardissono [1996]. This approach presents a cooperative dialog system which supports students of a computer science department. During the dialog the student's plan is identified and the interaction is adapted to the student's knowledge represented in the user profile. The adaptation to the user's level of knowledge is a capability from which other applications, e.g. adaptive help and E-Learning systems, may also profit.

The TOOT system allows the user access to online train schedules via speech dialog. An adaptive version of TOOT has been created which automatically adapts its dialog strategy to a changing user model [Litman and Pan 2002]. The user model is different from the above mentioned models in that it also indicates, whether the user is having speech recognition problems. The dialog behavior is then adapted accordingly.

For further information on dialog systems employing user models for improving dialog performance we refer to Zukerman and Litman [2001], where a detailed survey is given, and to Kobsa and Wahlster [1989], describing several of the earlier profile-based research dialog systems.

2.2.3.3 Using Dialog Systems for Collecting User Information and Offering Adaptive Services

Zukerman and Litman [2001] also point out advantages of systems that create and (dynamically) update user profiles through natural language processing, although they focus on the converse direction presented in the previous section. In this section, we discuss how approaches can use natural language interaction to acquire profile information and what the differences are in existing applications (cf. Figure 2.9, Case 2).

Jameson [2003] mentions two typical purposes for which adaptive systems can be applied. First, they can *support system use* (e.g. taking over parts of routine tasks, adapting the interface, giving advice about system use, controlling a dialog). Secondly, they are able to *support information acquisition* (e.g. helping users to find information, tailoring information presentation, recommending products, supporting collaboration and learning). As we will see in the sample approaches presented in this section, these tasks can in many cases effectively be supported by conducting a natural language dialog interaction with the user.

With respect to natural language profiling systems, we can distinguish between systems which *directly ask questions about specific preferences* (which is also the approach of this thesis) and systems which conduct a dialog with the user, but *learn profile information implicitly from the user's reactions and answers in this dialog*. Since the first type belongs to explicit acquisition and the second type to implicit acquisition, we also notice the same quantitative distribution here as mentioned earlier: implicit natural language profiling approaches outnumber explicit ones. We will see in the following what difficulties the two types of approaches have to address and how they can meet important requirements.

Type 1: Systems which ask about preferences directly

The first system type asks about preference information in a goal-oriented manner. The idea is to complete some basic user profiling before adapting services. This strategy enables a system-independent and reusable profiling approach.

A major problem which has to be considered in order to make this approach viable in practice, however, is that users do not get information immediately but have to wait until the profiling phase is completed. This could cause them to get impatient more easily, and for this reason the dialog duration should be kept as short as possible according to a given task. As is

pointed out by Kohlmaier et al. [2001], the visitor of a Website should not be harassed with too many questions. A possible solution to this problem is to investigate different possibilities for **selecting questions adaptively** (e.g. [Schmitt and Bergmann 2001, Stolze and Ströbel 2001]).

Stolze and Ströbel [2001] introduce the term *adaptive interviewing*, which denotes a strategy for selecting questions while at the same time asking as few questions as possible. They show, how a questioning strategy based on decision trees can be optimized. The problem with decision trees is that any change in the selling strategy or product usually requires a complex tree restructuring. The solution presented by Stolze and Ströbel [2001] allows generating optimized decision trees in a more efficient way, while simultaneously minimizing the user exit risk. They distinguish, for instance, questions with a high risk (the user might exit the interview and not buy anything) from those with a low risk and assign a probability to each question depending on previous questions and answers. The most important aspect is that *all* (not only a subset of two or three) further questions are searched for the one that fits best. Therefore, it is important that the questions do not depend on each other (i.e. they may be asked in any order). The method presented by Stolze and Ströbel [2001] is supposed to work even if no initial (sales) data about customers is available and if product information changes constantly. Schmitt and Bergmann [2001] point out that questions have to be selected depending on the situation and dialogs have to react flexibly to customer decisions. They argue, as do Stolze and Ströbel [2001], against a static order of questions and for dynamic question selection. They present an approach using “dynamically directed sales dialogs”, for which they formulate several requirements:

- The number of questions has to be kept as small as possible and only questions that help the system to gain and reuse experience may be asked.
- Questions have to be comprehensible for the customer. Therefore, different questioning styles should be provided for experts, novices, people with different levels of education, etc. Comprehension of questions can also be supported by clustering sequences of questions in a logical order.
- The answering cost for the customer must be kept low, i.e. questions that can be answered immediately must be preferred.
- The use of a dialog history helps the system to avoid repeating the same question.

Raskutti and Zukerman [1997] describe how questions can be selected during a *clarification dialog*, which may, for instance, occur if the system is uncertain about the user’s intentions or information is missing to identify these intentions. In their approach – implemented in the travel agent system RADAR – possible user intentions are connected with probabilities and are represented by means of an inference tree. Their mechanism determines the level in the tree at which the next question is to be directed. Branching points in the tree represent ambiguities, which may be asked about in the clarification dialog. If the information content of an interpretation of a user’s intent falls below a certain threshold, then further information needs to be acquired from the user. For this purpose RADAR offers two methods. In the first method, the dialog is kept as short as possible. The potential information content of a set of parameters is evaluated and the parameter with the highest assumed information content is chosen for the next question. The second method resembles and has been designed on the basis of real dialogs in the travel domain. It reflects certain habits, for example, that travel agents more often ask for the departure than for the arrival time, or that essential aspects of the trip are confirmed before acquiring details. An experiment investigates which method is most suitable in which situation (cf. [Raskutti and Zukerman 1997]).

Another approach which elicits user preferences directly is described by Felfernig et al. [2004]. Felfernig et al. [2004] investigate the integration of language processing techniques into the dialog design of knowledge-based configuration systems. The intended effects are more efficient dialogs, i.e. the user will need less clicks and receive a fitting configuration result faster. Their configuration system integrates a natural language based interface, which enables users to enter product preferences in their own words as shown in Example 2.2 (source: [Felfernig et al. 2004]):

Example 2.2:

“I’d like to buy a computer. The price of the computer must not exceed Euro 1500. The minimum capacity of the main memory must be 1024 MB. The CPU should at minimum have a clockrate of 2 Ghz. Furthermore, CDraw should be installed.”

The user preferences are extracted from the natural language input and are then transformed into the formal representation of a product configuration. The extraction of preferences is based on the assumption that the natural language input contains concepts (or synonyms of concepts) stored in a concept base or knowledge base. Domain specific chunks of words are searched for in the user input and matched against configuration relevant description fragments. The result of this process is either a configuration result, a follow-up question for further user information, or an explanation in case the user requirement cannot be satisfied because it would violate some product constraints.

Type 2: Systems which learn from dialog interactions

Some systems acquire profile information implicitly while they communicate with the user, for example, while providing the user with information or talking about products. During this communication session they analyze user preferences, discourse patterns and other information contained in the dialog, which helps them to update their assumptions about the user’s interests and intentions. In the following, we present examples of such applications.

Their BGP-MS user modeling shell system developed by Pohl et al. [1995] can support different applications in acquiring user information (i.e. knowledge or goals of the user). The profiling method is based on the BGP-MS capability to analyze dialog acts (based on speech acts, such as question, answer, agreement, rejection, etc.) during a natural language-based interaction. In order to make use of this capability, the application has to pass to BGP-MS among other things all dialog act types which are relevant for the application. For this purpose, a set of pre-defined and application-independent dialog act types offered by BGP-MS can be used. This set can be extended with one’s own dialog act types. During run time, the application provides information about dialog acts observed in the dialog, which are used by BGP-MS to maintain the user model [Pohl et al. 1995].

There are a variety of **recommender systems** using natural language interaction for various purposes [Felfernig et al. 2004, Wärnestål 2005]. [Chai et al. 2001], for example, describe a natural language sales assistant which helps users find relevant information about products and services in E-Commerce sites. This assistant software analyzes consumer requests in natural language to customize Website navigation or to give product recommendations. Customers can, for instance, specify the type of product or product properties (price, color, weight, etc.) they are looking for. The dialog manager arranges follow up interactions asking for more information, providing feedback and confirmation, etc. The actual profiling takes place during these assistance dialogs. Both customer and business requirements can be accommodated by combining traditional rule-based technology from artificial intelligence with taxonomy mapping [Chai et al. 2001]. The online interaction subsystem of this approach (including the dialog manager) is described as domain independent.

Not all systems acquire profile information *while* the natural language interaction takes place. Some approaches learn user preferences from data stored in log files *after* the interaction with the user. An example for such a system is COGITO [Abbattista et al. 2002b]. In the COGITO scenario, the user is questioned by a chat bot (cf. Section 2.2.3.1) of an online bookstore about the books or authors she is looking for. The virtual assistant provides access to presumably relevant product information, such as book and CD descriptions, or helps the user find appropriate offers. The conversation is stored in a log file to which a machine learning algorithm is applied after the conversation, in order to learn about specific book categories that are of interest to the user. This information is used for *future interactions* with the same user. The information helps to improve searching in the product database (e.g. by dissolving search ambiguities in user queries) so that the virtual agent can make adequate, personalized book recommendations.

Figure 2.10 shows a conversation between the COGITO chat bot Susanna, which has been integrated into the Website of the online book vendor BOL (www.bol.de), and a new customer. At the bottom of Figure 2.10, the search results (books of the author “King”) are listed. These search results can be refined with further information about the user’s preferences, e.g. that the author “King” she is looking for mainly writes computer science books:

Susanna: Good evening! Nice to see you found your way to BOL.DE. May I introduce myself? My name is Susanna and I am your personal assistant at Bertelsmann Online. What is your name?

User: My name is Fabienne.

Susanna: Nice to meet you, Fabienne. What can I do for you?

User: I'm looking for a book.

Susanna: Then this is the right place to be! We have hundreds of thousands of deliverable books. Are you looking for a book by a specific author?

User: Yes

Susanna: What is the name of the author?

User: King

Susanna: I shall check whether we have a book by the author King. Please wait ...




Figure 2.10: Dialog between COGITO chat bot Susanna and new customer with succeeding book recommendation (source: [Abbattista et al. 2002b])

Strictly speaking, the COGITO system is a mixture of Types 1 and 2: the chat bot asks users about the objects they are looking for (e.g. types of books, genres, or authors), i.e. it elicits information directly (Type 1). This information is only evaluated after the dialog interaction, however, by means of implicit methods (Type 2). This makes the approach vulnerable to the *Cold Start Problem* (cf. Section 2.1.1.4) [Abbattista et al. 2002a], i.e. the system cannot profit from user data during the first encounter with the user, but only in follow-up interactions. This may lead to poor recommendation quality at the beginning, which has to be compensated

for by other methods. Possibilities for compensation may be, for instance, having the bot chat with a user “just for fun” and to collect user preferences during this chat or applying the learning algorithm during (rather than after) the dialog interaction [Abbattista et al. 2002a].

In the Adaptive Place Advisor [Göker and Thompson 2000], the recommendation of a destination, such as a restaurant, is an interactive, conversational process during which the system builds a user model implicitly. The user model contains various types of information extracted during the dialog, for instance, preferences about specific items, attributes, or values, the relative importance of them, desired combinations, etc. Item preferences, for example, are learned by observing how often a suggested item was accepted or rejected by the user. The sequence of questions asked by the system has the main goal of eliminating items from consideration, in such a way that a small set of recommendable items finally remains.

Another system for dialog-based restaurant recommendations is MATCH (Multimodal Access To City Help) developed at AT&T Labs [Johnston et al. 2001]. This approach provides information about New York City restaurants and subway by offering an integrated speech and pen interface for multimodal interaction. It may either run in client-server mode or as standalone version, which enables its use even if no wireless data network is available. The system does not only take into account user preferences stated in the dialog, but also information about the user’s physical environment, as well as changing tasks at hand. It includes machine learning techniques for simultaneous integration of multiple modalities (e.g. a user circles an area on a virtual map and says “show me restaurants in this area”).

In the interdisciplinary research project Forsip (www.forsip.de), new possibilities for human computer interaction for various applications (e.g. high-tech homes or personalized virtual Internet assistants) are investigated. Natural language interaction, multimodality, as well as personalization play a key role here. In the subproject COSIMA^{B2B}, user preferences are elicited during a natural language dialog with a virtual (sales) assistant. The assistant is capable of composing user queries in a personalized manner. For this task it uses preferences entered explicitly by the user, long-term user interests which match the actual situation, and preferences of the vendor. Long-term interests are detected by means of preference mining algorithms [Kießling 2005].

2.2.4 The Profile Acquisition Method Developed for this Thesis

In Section 2.1.2, we gave reasons for our Profile Model. It constitutes the basis for the profile acquisition method we have developed for this thesis: an adaptive natural language speech dialog for acquiring user interests. In Chapter 1, we have explained the fundamental ideas of our profiling method and why it is particularly suitable for an in-vehicle event and tourist recommendation scenario. We will now show how our profiling method relates to other existing approaches and in which respects it differs. Again, the focus of this section is on giving reasons for the general principles and design decisions which distinguish our approach from others, whereas Chapters 4 and 5 provide a detailed description of our profiling technique.

In Section 2.2.1, we discussed challenges for user profiling. A main challenge is to achieve maximal profile quality while keeping user effort as low as possible. A specific characteristic of a high-quality profile is that its contents are reliable, i.e. they actually match the user preferences to a high degree. The profile quality in most cases is directly mirrored by the quality of personalized recommendations generated. For this reason we decided to take an **explicit approach** to user profiling rather than merely observing user behavior. Our solution thus tries to overcome the limitations of implicit user profiling with regard to profile quality and reliability (cf. Section 2.2.2.1).

Our approach is **compatible with other profiling methods**, however, if this is desired. Although it can be used exclusively for in-vehicle profiling, for example, it can also be combined with methods which are typically not applied in this scenario (e.g. online question-

naires) or with implicit profiling. As discussed above, a combination of different methods which are specialized for different parts of profile information (e.g. questionnaires for basic personal data, dialog for complex interests, etc.) seems promising, especially for maintaining long-term profiles.

As argued in Chapter 1, mobile application situations like in-vehicle profiling and recommendations require an alternative, speech-based explicit profiling method, which we present in this thesis. However, **other application situations** (e.g. profile-based product recommendation on the Internet) can also profit from improved explicit profiling methods [Stegmann 2005], which becomes particularly clear if one considers the drawbacks of existing explicit methods discussed in Section 2.2.2.2.

Since personalization is a central topic for recommender systems and becomes increasingly important for other applications, especially in E-Commerce and Internet, we have designed a profiling method which meets this development. We create not only a truly **personalized profile** in that we acquire new, not predefined individual information by means of the lexical-semantic network GermaNet (cf. [Stegmann and Wörndl 2005]). Our **profiling technique is personalized** itself – and in this respect exceeds typical existing approaches such as static questionnaires – in that we adaptively tailor questions to interests the individual user has mentioned. We could not find a comparable explicit profiling approach that shows these characteristics. The profile-based recommender system of Chai et al. [2001], for example, which also employs a domain-independent profiling dialog approach, does not construct and store a detailed, semantically rich user profile comparable to ours. We assume that personalizing the profiling method itself is also an important criterion for finally increasing user loyalty. A dialog which adapts to each user personally is more likely to be perceived as more interesting, entertaining, and worthwhile communicating with than a static questionnaire, for instance, which looks alike for all users (also cf. discussion of our evaluation results in Section 6.3.3).

In this personalization aspect we are similar to **adaptive questionnaires**, because we primarily ask questions which are related to the interests entered by the user (i.e. we adapt the questions to the individual user). In this respect we also differ from **static questionnaires**, where the user has to answer the entire set of (standard) questions (no matter whether adequate or not). In consequence, we may gain more personalized information in the same or less time than is possible with a static questionnaire (cf. [Stegmann 2005]). We differ from adaptive questionnaires in two other respects, however. The adaptive questionnaire approaches we investigated are either tailored for specific application domains or stop as soon as the user can be assigned to a stereotype or user group. Our adaptive question selection can be used for various applications that want to acquire user interests. Moreover, the aim of our dialog is to collect as much individual information as users are willing to give rather than to assign them to a user group in the first place [Stegmann et al. 2004]. The assignment to a target group should rather be considered a secondary result of our approach and can, for instance, be used for marketing purposes. In this respect, we also differ from existing **stereotype or target group model** approaches (e.g. Lifestyle Finder [Krulwich 1997]). Our target group model fulfills the purpose of helping select meaningful questions, but our aim is not to take over a generalized group model as the actual user profile [Stegmann 2005].

Above we saw that in most approaches profiling and personalization methods are strongly intertwined, which makes it difficult or impossible to reuse the profiling part for other applications. The COGITO approach [Abbattista et al. 2002b], for example, is tailored to the domain of book recommendations and the profile attributes learned are a limited number of book categories, which occur in the product database of the Internet vendor. The approach of Raskutti and Zukerman [1997], for instance, offers different dialog strategies that can be chosen according to the dialog situation, but at least one of them is modeled for a specific domain (dialogs with a travel agent). Raskutti and Zukerman [1997] claim that the strategies can also be applied to other domains, but admit that it would be necessary then to investigate anew

which strategy is suitable for which situation. In our approach, the **dialog strategy does not depend on a specific domain**, but only on the profile contents (personal interests). Our profiling method is not bound to a particular adaptive system and is not part of the recommendation process. The profiling happens *before* the adaptive service takes part (e.g. recommendations are generated). This allows us to make the profiles accessible as long-term information for different applications.

In Section 2.2.3.3, we have discussed dialog profiling systems which **ask for information directly** and others which learn from a dialog interaction implicitly. Our approach belongs to the first category. Although it does not use decision trees, it is similar to the approach of Stolze and Ströbel [2001] in that we use a set of independent question frames which can be asked in any order and are selected individually for each dialog situation. As demanded by Schmitt and Bergmann [2001] and Stolze and Ströbel [2001] we enable a dynamic question selection in order to react flexibly to the interests stated by the user. As suggested by Schmitt and Bergmann [2001] and is common in many dialog systems, a dialog history helps our system to avoid repeating identical questions. Similar to Stolze and Ströbel [2001], our approach also does not depend on a set of initially available user profiles and can be employed right from the beginning. This also distinguishes it from machine learning-based approaches which may suffer from the Cold Start Problem (e.g. COGITO [Abbattista et al. 2002b]).

With respect to the language processing approach, we decided against using a **chat bot** (cf. Section 2.2.3.1) for the following reasons. We could not find a chat bot system which accomplishes a dialog or user profiling interaction at a satisfactory technological level. Chat bots are often not accepted as dialog partners by users, because it is only too obvious that they merely simulate a dialog instead of really understanding what the customer has said. Therefore, we favor a less pretentious and fragile interaction: we merely allow a system-initiative dialog which is able to accomplish the profiling task in a reliable manner (cf. Section 3.7.3). Moreover, as described above, chat bots typically require considerable manual effort for changing their language database in case the product or domain changes, which is not what we intended.

Furthermore, we do not use one of the existing more **complex dialog approaches** described in Section 2.2.3.3, for three main reasons. The first reason is again their dependency. Most of these dialogs are designed more or less for a specific application or domain and would not have been transferable to the task of acquiring user interests without considerable effort, if possible at all. Secondly, many of the above mentioned approaches follow a much broader and complicated dialog strategy, which allows for the recognition of different speech acts on the user's side as well as changes in initiative between system and user. Moreover, the dialogs often also offer the possibility of not only requesting but also presenting information to the user, which is more than we need. With our decision against these approaches we wanted to avoid technical redundancy and unnecessary complexity, which might have caused further problems. Thirdly, we needed a dialog strategy which specializes in the acquisition of user interests. Since there are no comparable systems which acquire semantically complex interests similar to ours by means of a natural language dialog, we had to design our own dialog strategy. The choice of our dialog strategy is further justified when discussing dialog system technologies in Section 3.7.3.

2.3 Summary

In this chapter, we have presented fundamental and related work in UM and profile acquisition. We showed that existing approaches to UM are mostly application-dependent and reusability for different domains and scenarios often is an unsolved problem. Moreover, most existing adaptive systems, such as recommender systems, use implicit user models in order to

reduce effort on the user's side. This circumstance is not only connected with the problem of transparency, user cooperation, and trust, but also with profile quality and hence the quality of the adaptive services themselves, which are based on these profiles.

We discussed advantages and disadvantages of implicit and explicit profile acquisition and the characteristics of profiling approaches involving natural language. Implicit approaches have to cope with problems such as reliability, misinterpretations of user actions, and the fact that not all types of information can be acquired implicitly. But also explicit profiling approaches would clearly profit from improved methods. They struggle with overstrained users who balk at rating too many items or filling in tiresome standard questionnaires. Natural language-based approaches, on the other hand, suffer from the drawback that interaction with them can be time consuming and that adaptive dialog systems, which react flexibly to the user's needs, are difficult to develop.

On the basis of these considerations, we described the main requirements and features of our Profile Model: it models complex user interests entered in natural language, is flexible enough for storing not predefined input, relates the acquired information semantically, is reusable, and profiles are kept over multiple sessions. A target group model is only used for dialog control, not for taking a rather general group characterization as profile for an individual user. Our explicit profiling method is compatible with implicit profiling or questionnaires. It is tailored to the acquisition of user interests, but this does not restrict its applicability: it can be applied in many domains or scenarios in which interest information is relevant. Our Profile Model has been designed to meet the requirements of our specific application scenario, where user information has to be acquired by means of speech dialog, but the intention was also to overcome some of the drawbacks of existing explicit profiling approaches mentioned above. We also explained that, in our opinion, personalization also requires personalized profiling methods. Our profiling method is personalized in that we adaptively tailor questions to interests the individual user has mentioned. The question selection and topics discussed differ for each user. The profiles created with this method are truly individual in structure and contents.

3 Natural Language Processing Methods and Resources Applied in this Thesis

In this chapter, we describe which resources and methods from the area of human language processing are applied in our solution. We explain for which tasks they are used as well as their advantages and limits with respect to 1) the goal of acquiring profile information from natural language user input, 2) semantically relating the acquired information items to each other in a dynamically changing user profile, and 3) generating questions for the user. These tasks are described in detail in Chapters 4 and 5.

Chapter 3 is structured as follows. We first give a brief introduction to natural language processing and explain how it relates to user profiling in Section 3.1. The remainder of the chapter is dedicated to specific fields in the area of natural language processing. Since we argue that our solution can be applied not only for typed input/output but also with a speech interface for in-vehicle use, we address the topic of speech recognition and synthesis in Section 3.2. We then describe part-of-speech tagging (3.3) and syntactic analysis (3.4) of user answers. In Section 3.5, GermaNet is introduced, which is employed for constructing and manipulating the profile graph. Since natural language user input may contain ambiguities, Section 3.6 discusses the topic of word sense disambiguation. In Section 3.7, existing (speech) dialog systems will be addressed. Since our approach aims at generating questions adaptively and individually for each user, natural language generation is finally the topic of Section 3.8. It would exceed the scope of this thesis to discuss all of these comprehensive research fields in full detail. Therefore, each of the Sections 3.2 to 3.8 is structured as follows: we first give a short overview of the research field and then focus on the special technologies in this area which are used for or play a role in the LINGUINI system.

3.1 Applying Natural Language Processing for User Profiling

In this section, we introduce central topics, tasks, and methods of natural language processing. However, we cannot completely cover this large field here, but rather concentrate on aspects which are necessary for understanding the approach of this thesis. Detailed introductions are given, for instance, by Carstensen et al. [2004] or Jurafsky and Martin [2000].

The interdisciplinary research area of *computational linguistics* investigates how human language can be formalized and processed by computers. It is therefore also referred to as *natural language processing (NLP)*. Theoretical and technological influences come from various areas, such as computer science, linguistics, cognitive psychology, mathematics, or electrical engineering [Jurafsky and Martin 2000]. The representation of knowledge about human language is one of the central topics in NLP. Natural language can be investigated at several levels. Jurafsky and Martin [2000] mention the following commonly assumed levels (also cf. [Grewendorf et al. 1987, Carstensen et al. 2004]):

- “Phonetics and phonology – the study of linguistic sounds
- Morphology – the study of the meaningful components of words

- Syntax – the study of the structural relationships between words
- Semantics – the study of meaning
- Pragmatics – the study of how language is used to accomplish goals
- Discourse – the study of linguistic units larger than a single utterance”

[Jurafsky and Martin 2000]

Various types of frameworks, formalisms, and methods have been developed for each level. Most natural language software applications (e.g. machine translation systems, dialog systems) consist of a combination of methods and models from the different levels [Carstensen et al. 2004]. According to Jurafsky and Martin [2000], the most commonly used technologies in NLP are what the authors call “state machines” (e.g. finite-state automata and transducers, Markov Models), formal rule systems (e.g. regular, context-free, feature-augmented grammars), logic (e.g. predicate calculus), probability theory, and machine learning. In the remainder of this chapter, we will further describe several of these technologies.

NLP systems are applied in any scenario where human language is the most suitable and natural way of interacting, handicapped users need to be supported, speech input is required, or written texts (e.g. documents on the Internet) need to be automatically translated, summarized, categorized, searched, etc. As has been pointed out in Chapters 1 and 2, a particularly NLP-compatible and strongly growing field is **personalization**. Zadrozny et al. [2000], for instance, explain the importance and possible benefits of using natural language dialogs in this field. They argue that NLP can be a supporting technology for personalization, because users can interact with the system in their own words, i.e. the interaction itself is personalized. **E-Commerce**, which is also strongly influenced by personalization trends, in general seems to be a highly suitable application field for natural language systems. An evaluation conducted by Chai et al. [2001] shows that 79 % of users prefer a natural language-based navigation over a menu-driven navigation (21 %) in the E-Commerce context.

On the basis of these observations, it seems promising to also apply NLP for **user profiling**. The benefits of creating and dynamically updating user profiles through NLP have often been pointed out (e.g. [Zukerman and Litmann 2001]). However, no approaches seem to exist, which use NLP methods and resources for dynamically creating and maintaining a semantically rich profile comparable to the one developed for this thesis, in combination with an application-independent profiling dialog which is specialized in the acquisition of personal interests.

In our approach, we want to give more freedom to users by admitting answers formulated in their own words. Interpreting free natural language answers, however, is a complex process to which several NLP technologies contribute. Figure 3.1 below illustrates the tasks of the NLP methods and resources involved in our profiling process. They can be assigned to four areas: answer analysis, profile management, dialog management, and question generation:

- **Answer analysis:** The spoken user answer has to be transformed into an interpretable character string (speech recognition). Then, this string is linguistically analyzed (tagging and parsing). Ambiguities in the user input have to be resolved (word sense disambiguation).
- **Profile management:** Information items extracted from the user answer have to be inserted into a semantically structured profile (lexical-semantic network).
- **Dialog management:** The interaction with the user has to be controlled, the contents of questions have to be selected (dialog system).
- **Question generation:** The final question string has to be generated (natural language generation) and transformed into an acoustic signal (speech synthesis).

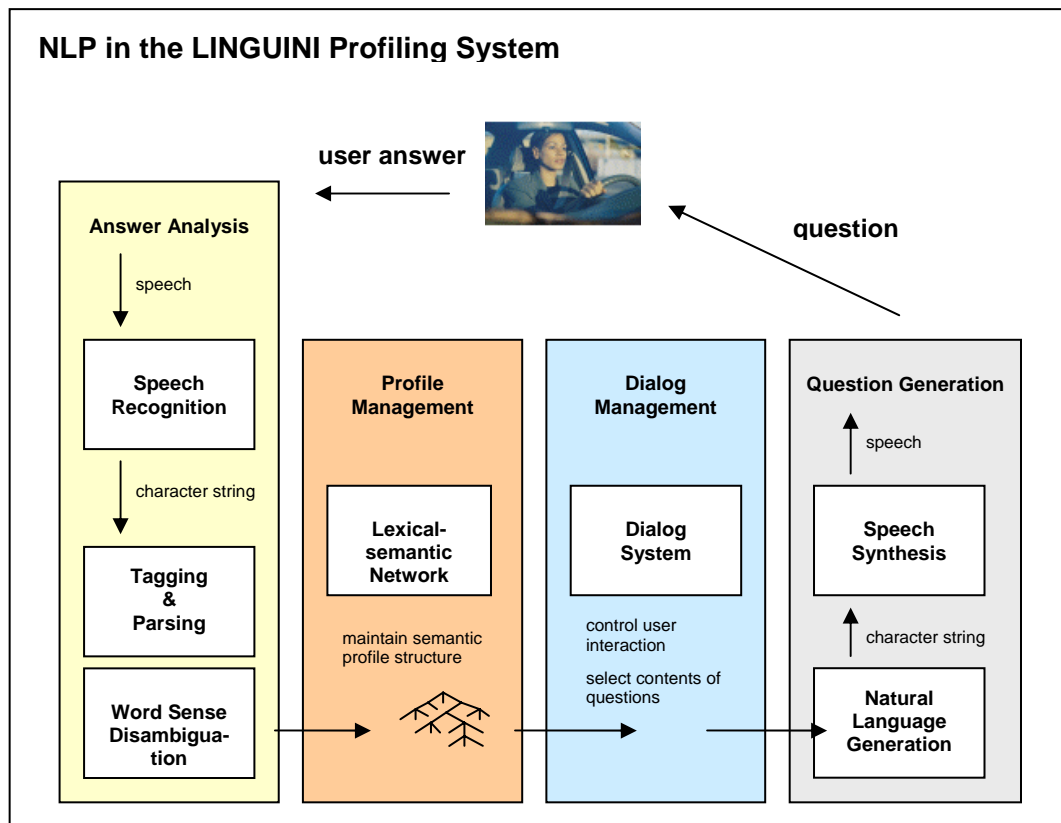


Figure 3.1: Natural language processing tasks in the LINGUINI profiling system

In the following, we will discuss each of these tasks in detail.

3.2 Recognition and Synthesis of Spoken Language

A speech interface typically has to accomplish two tasks. On the one hand it performs *speech recognition*, i.e. it recognizes an acoustic speech signal and creates a character string representation for it, which can be further processed by other NLP components. This transformation process is also called *speech-to-text*. On the other hand, it has to transform a system utterance into an acoustic signal which can be understood by the user. For this purpose it can use *speech synthesis* or prerecorded speech [McTear 2002]. This direction is called *text-to-speech*.

3.2.1 Speech Recognition (Speech-to-Text)

The speech recognition task typically consists of the following steps [Zue and Cole 1997]. First, the digitized speech signal is transformed into a set of useful measurements or features at a fixed rate. For this purpose the speech waveform is processed into a vector representation which contains specific parameter values. The parameters may be used to estimate the probability that this waveform corresponds to a particular phonetic event (cf. [Hunt 1997]). These measurements are then used to search for the most likely word candidate. A speech signal which consists of more than one word can be analyzed by means of language models or grammars. These help to restrict the number of possible combinations of words. Often, finite-state networks are used for this purpose, where a sequence of states defines an allowed se-

quence of words. Furthermore, so-called acoustic models (e.g. based on Hidden Markov Models) and also lexical models can be used (cf. [De Mori and Brugnara 1997]).

There are speech recognition systems which are speaker-independent and systems which require a speaker “enrollment” by giving speech samples [Zue and Cole 1997]. The latter are capable of recognizing a speaker by individual features included in her speech waves. This is useful for a variety of purposes, such as enabling controlled access to computer rooms, telephone banking and shopping, voice mail, or the access of confidential information services. Speaker recognition can be divided into speaker identification and speaker verification [Furui 1997]. During *speaker identification* it is determined which registered speaker provides the utterance at hand. *Speaker verification* is the process of accepting or rejecting the identity claimed by a speaker.

Speech recognition is still a difficult task. This is due to a set of variabilities associated with the speech signal [Zue and Cole 1997] (also cf. [McTear 2002]):

- **Phonetic variabilities:** The pronunciation of the smallest sound units differs according to the linguistic context in which they occur.
- **Acoustic variabilities:** Changes in environment and location and various types of background noise can influence the recognition process.
- **Within-speaker variabilities:** Physical or emotional states, speaking rate, or voice quality of a speaker may change.
- **Across-speaker variabilities:** Between speakers, sociolinguistic background, dialects, and vocal tract size and shape may differ.

The recognition of an isolated word is much easier than the recognition of continuous speech [Zue and Cole 1997]. The recognition of spontaneous speech poses additional problems, such as word repetitions, interjections and disfluencies, disrupted sentences and corrections, and the like. Disfluency, for example, can be addressed by introducing explicit acoustic models for filled pauses [Zue and Glass 2000]. Ward [1989] has shown that system performance can be increased by modeling noises, which are caused by speaker or environment. Here, the idea is to train Hidden Markov Models with “noise words”, so classes of noises can be detected. Several of the above mentioned problems, e.g. the robust recognition of spontaneous speech, the handling of self-corrections, or multilingual speech recognition, have been investigated in the VERBMOBIL project [Wahlster 2000].

Problems furthermore occur, if utterances contain words outside the system’s vocabulary. Most current systems have been developed for restricted domains with specific vocabularies [Zue and Glass 2000]. If the system vocabulary is very large, such that users cannot remember its complete contents, they will very likely use words outside this vocabulary [Asadi et al. 1989]. This problem is typically not reported back to the user, because the system recognizes some other (wrong) word from the vocabulary instead. There are several approaches which try to deal with the handling of unknown words. Asadi et al. [1989], for instance, have developed an acoustic model for unknown words, with which they achieve a detection rate of 74 %. After a new word has been detected, it can be added to the vocabulary.

The following speech recognition systems are freely available for research purposes. The CSLU Speech Toolkit (cslu.cse.ogi.edu/toolkit), for example, offers a comprehensive suite for speech recognition, synthesis, and dialog. The Mississippi State ISIP public domain speech recognizer (www.cavs.msstate.edu/hse/ies/projects/speech/index.html) also includes a tool for Hidden Markov Modeling and decision trees. The Galaxy Communicator developed in the DARPA Communicator program (communicator.sourceforge.net) is an open source architecture, which enables the construction of dialog systems by combining architecture-compliant commercial software and research components. A more recent DARPA supported project

conducted at the University of California at Berkeley is “Effective Affordable Reusable Speech-to-text (EARS)”, which aims at recognizing broadcast news and human-to-human conversational speech. Another large project, conducted by universities and industrial researchers in the late nineties, was DISC. The DISC Website (www.disc2.dk) provides an overview of speech recognition, synthesis, and dialog systems in various publications and state-of-the-art reviews.

Main commercial applications are, for example, IBM’s “ViaVoice” or the Nuance (www.nuance.com) product “Dragon Naturally Speaking 8”. Moreover, Philips offers a range of speech recognition products (www.speechrecognition.philips.com). These systems are also known as *dictation software*, which enables the creation of texts, letters, or emails without using a keyboard. Their vocabulary coverage is rather large: the IBM “ViaVoice Pro USB Edition” for German, for instance, contains approximately one million words. Unknown or domain specific words can be added to the vocabulary in addition. Some of these systems can also be trained for a particular speaker’s voice, for instance by having the speaker read some training texts at the beginning, which improves correctness of the recognition. IBM and Nuance are also main suppliers for speech recognition software for the automotive domain, in addition to Temic SDS (www.temic.com). Today, speech recognition systems are used by almost all major car manufacturers for high-end cars to support the use of various in-vehicle functionalities. This topic is discussed in detail in Section 3.7.2.

3.2.2 Speech Synthesis (Text-to-Speech)

In general, spoken output can be generated in two ways. First, the system can use prerecorded, “canned” sentences and phrases which are merely selected or combined to form the desired output [McTear 2002]. The *prerecorded speech* may contain slots which can be filled with other prerecorded samples. The second, technically more complex possibility is to use *speech synthesis*. Speech synthesis is required in case the output is variable and unpredictable, when the output is a selection of a larger text, or when consistency of voice is required [McTear 2002]. According to McTear [2002], speech synthesis consists of two tasks: text analysis (also called text-to-phoneme conversion) and speech generation (also called phoneme-to-speech conversion). During *text analysis*, the text is analyzed and stored in a linguistic representation. This representation will be used for producing synthetic speech. The text analysis phase can be subdivided into several tasks, such as segmentation and normalization of the input, morphological analysis, syntactic tagging and parsing, and the modeling of continuous speech effects in order to create a naturally sounding continuous speech output. During the *speech generation* phase, a prosodic description including rhythm and intonation is created and the final waveform is synthesized. In this phase, different types of models, e.g. *articulatory synthesis* or *formant synthesis*, can be used [McTear 2002]. The most natural sounding output is produced by systems involving *concatenative synthesis*, where prerecorded units of speech are selected and joined together in speech generation.

In parallel to speech recognition systems, there are a number of research prototypes and commercial text-to-speech systems available. An open source synthesis system for English and Spanish has been developed in the FESTIVAL project at the University of Edinburgh (www.cstr.ed.ac.uk/projects/festival). It uses *diphone synthesis* (a subtype of concatenative synthesis). A German version of FESTIVAL is available at the University of Stuttgart (www.ims.uni-stuttgart.de/phonetik/synthesis). The diphone synthesis technique is also applied in the SVox system for German (people.ee.ethz.ch/~svox/cgi-bin/w3svox). The SVox research project conducted at ETH Zürich has concentrated on prosody control, morphological, and syntactic analysis. Another synthesis software which is free for research purposes and covers dozens of languages is MBROLA (tcts.fpms.ac.be/synthesis). A well-known commercial system is Laureate by British Telecom. It employs another type of concatenative method

called *unit selection*. AT&T has developed Natural Voices™ (www.naturalvoices.att.com), which supports various languages including English and German.

Speech synthesis in the vehicle is a rather difficult task. One reason is that existing synthesis systems in general need large amounts of memory in order to create good intelligibility, which may exceed the limits of telematic systems. Special synthesis systems for in-vehicle use have, for instance, been developed by Nuance, SVOX, and ELAN. The product “acapela onboard” by ELAN (www.elantts.com) is a synthesis system adapted to the constraints of the automotive industry by means of special compression techniques. IBM’s ViaVoice has been used for text-to-speech in Honda vehicles and supposedly produces a highly natural-sounding output. Further synthesis systems as well as sound samples can be found on a Website of the University of Stuttgart (www.ims.uni-stuttgart.de/~moehler/synthspeech/examples.html).

3.2.3 Speech Recognition and Synthesis in LINGUINI

In the adaptive dialog profiling concept developed in this thesis, a speech recognition and synthesis module are envisioned for enabling a dialog in spoken language. The tasks of these modules in the profiling process have been explained in Section 3.1 (cf. Figure 3.1). We have, however, not yet physically integrated a speech user interface into our prototype implementation. Our focus was on realizing the main functionality of the profiling system to show its general feasibility. However, we think that a speech interface may be integrated in general and will briefly sketch the specific requirements our system imposes on this interface.

Requirements for the Speech-to-Text Component

The vehicle scenario imposes some restrictions on the speech recognition component. It has to be able to process continuous speech and must be robust enough to cope with all sorts of background noise. Moreover, since we want to acquire individual profiles by means of free user input, the vocabulary size has to be comparable to that of dictation systems. It also has to be considered that in our system the user input phrases are rather short. This might be a problem for some speech recognizers, since only a small amount of word context is provided for determining the most likely word candidate.

Considering these circumstances, the speech recognition component of LINGUINI could be realized by using a speech-to-text system for German comparable to ViaVoice for example (cf. Section 3.2.1). The speech recognizer should be trained on the background noise which occurs in a vehicle and, if possible, also for the particular user’s voice to achieve better results. We also recommend integrating a feedback mechanism with which the user can enter new words the speech recognizer cannot handle yet.

Requirements for the Text-to-Speech Component

In dialog systems, little emphasis has been put on the generation of spoken output in general, because text-to-speech systems are commercially available that produce reasonably intelligible output [McTear 2002]. For the same reason, we did not focus on this part, but assume that an existing speech synthesis system can be applied here. Our profiling system uses question frames which contain variable slots. We would argue for the use of concatenative speech synthesis (cf. Section 3.2.2) for achieving a more naturally sounding result than is possible with other methods. This approach should be applicable here, because the question frames we use could be prerecorded and would only have to be filled with words relevant to the current interaction. In addition, continuous speech effects might be modeled to smoothly integrate the pronunciation of the slot-filling words with the pronunciation of the frame.

3.3 Part-of-Speech Tagging

3.3.1 Taggers and Tagsets

Since our profiling method cannot only be applied for spoken language but also for text input, we address the topic of part-of-speech tagging in this section. Analyzing natural language user input involves several processing steps. A step which normally precedes syntactic analysis (cf. Section 3.4) is *part-of-speech tagging* (short: *tagging*). The aim of tagging is to assign a part-of-speech tag to each word occurring in a text or user input. Tools that perform this task are called *part-of-speech taggers* (short: *taggers*). A *part-of-speech* can be regarded as the morpho-lexical category of a word. It expresses information about the lexical, morphological, semantic, or distributional properties of words [Schiller et al. 1995]. Part-of-speech tags indicate, for example, whether a verb occurring in a specific position of a sentence is a full verb or an auxiliary or whether a word is a noun or an adjective and whether the adjective occurs in a position before or after the noun. The specific part-of-speech tags used by a tagger are defined in the *tagset*.

Word Class Disambiguation

The problem taggers have to solve is **word class** or **part-of-speech disambiguation**: a specific lexical word form may have different part-of-speech tags in different contexts. Figure 3.2 illustrates this problem:

the	can	will	rust
det	modal-verb	modal-verb	noun
	noun	noun	verb
	verb	verb	

Figure 3.2: Possible parts-of-speech in order of frequency (source: [Charniak 1997])

Under each word in the sentence “the can will rust”, possible part-of-speech tags are listed in order of frequency. The word “can”, for instance, may be a modal-verb, a noun, or a main verb in different syntactic contexts. So the tagger has to take into account the word context in order to make the right choice. The correct choices for this sentence are marked in bold type.

Tagsets

Tagsets differ according to the natural language that is supposed to be tagged and according to the desired depth and type of annotation. Part-of speech tags may be simple, such as “verb”, “adjective”, etc. or may contain additional information, such as “VVFİN” meaning “Vollverb, finit (finite main verb)”. The tagger usually places tags behind the word form, e.g. “schwimmt/VVFİN (swims/VVFİN)”.

A tagset frequently used for English is the Penn Treebank tagset [Mitchell et al. 1994], which is a simplified version of the Brown Corpus tagset and contains 36 part-of-speech tags and 12 other tags for punctuation and currency symbols (as opposed to 87 tags used in the Brown tagset) (cf. Figure 3.3):

1. CC Coordinating conjunction	25. TO <i>to</i>
2. CD Cardinal number	26. UH Interjection
3. DT Determiner	27. VB Verb, base form
[...]	
7. JJ Adjective	31. VBP Verb, non-3rd ps. sing. present
8. JJR Adjective, comparative	32. VBZ Verb, 3rd ps. sing. present
9. JJS Adjective, superlative	33. WDT wh-determiner
[...]	
12. NN Noun, singular or mass	36. WRB wh-adverb
13. NNS Noun, plural	37. # Pound sign
14. NNP Proper noun, singular	38. \$ Dollar sign
15. NNPS Proper noun, plural	39.. Sentence-final punctuation
[...]	

Figure 3.3: Cutout of the Penn Treebank tagset (source: [Mitchell et al. 1994])

Another English tagset is the C5 tagset used in the CLAWS project to tag the British National Corpus (BNC) [Garside et al. 1997].

For German, a standard tagset is the Stuttgart-Tübingen-Tagset (STTS) described by Schiller et al. [1995]. The STTS contains 54 tags at a comparable level of detail as the Penn Treebank tagset. For further information on English word classes and tagsets cf. [Jurafsky and Martin 2000].

Tagging Approaches

According to Abney [1996a], *rule-based taggers* and *stochastic taggers* (or *Hidden Markov Model (HMM) taggers*) can be distinguished. Jurafsky and Martin [2000] mention *transformation-based taggers* in addition to these.

Rule-based taggers assign tags by using a lexicon and morphological analysis. Unfitting tags are eliminated on the basis of conditional rules, which check whether specific contextual requirements are met. They are able to perform equally well or better than stochastic taggers, but this can only be achieved by investing considerable manual effort in writing rules [Abney 1996a]. Since most rule-based taggers are deterministic (unlike stochastic ones), they are also very fast. A sample rule-based tagger, EngCG, is described in [Jurafsky and Martin 2000].

Stochastic taggers calculate the probability of a given word having a specific tag in a given context. They are typically trained with a text corpus and use a Hidden Markov Model (HMM) approach. The probabilities are derived from a training corpus. It is important that the contents of the corpus are carefully chosen and that the corpus is also large enough. Originally, HMM taggers operate on bigrams, i.e. that the probability of a tag depends on the previous tag. Today, many HMM taggers use an extended approach to achieve higher accuracy, which is based on trigrams, i.e. the two previous tags [Jurafsky and Martin 2000]. An example for a stochastic trigram tagger is “Trigrams’n’Tags (TnT)” [Brants 2000]. TnT is based on the Viterbi algorithm for second order Markov models. It is optimized for speed, can be trained for different languages and tagsets, and integrates methods for smoothing and handling unknown words. Detailed information about stochastic taggers can, for instance, be found in [Manning and Schütze 1999].

A famous example of a *transformation-based tagger* is the Brill tagger [Brill 1995], which shares features of both above mentioned approaches. Like rule-based taggers, it uses rules for determining the tag of an ambiguous word, but it also has a machine learning component,

which learns these rules automatically from a tagged training corpus [Jurafsky and Martin 2000].

3.3.2 Choosing a Tagger for LINGUINI

Tasks and Problems

In our approach, a tagger is used for analyzing the answers a user gives to our profiling questions. After the tagger has assigned the correct part-of-speech tags to the user input, it can then be further processed with a parser (cf. Section 3.4).

With respect to the type of user input given in our system, we found that the following phenomena may cause problems for a tagger:

- Misspelled words
- Unknown words
- Proper names

If our system is used with typed input, **typing errors** may occur. Words which contain typing errors are difficult to recognize for a tagger. A solution is to have the user input checked and corrected by a spell checker, i.e. a tool for eliminating spelling mistakes, before applying the tagger. A well-known approach to spell checking is the Levenshtein algorithm. The algorithm is explained, for instance, in [Kruskal 1999]. At the moment, our prototype does not contain a spell checker, but may be extended to contain one in the future. Presently, LINGUINI expects correctly spelt user input.

Another problem is the occurrence of words which are spelt correctly, but are nevertheless **unknown** to the tagger, because they did not occur in the training set. This is also often the case for foreign language expressions. For this reason, taggers are usually trained for the particular type of input which is expected in the application domain. Training requires large amounts of training data, however. Since we did not yet have any user interests or profiles for training the tagger, we had to find a tagger which has already been trained on a large, comprehensive data set. In addition, we ask users to avoid foreign language expressions if possible for the time being (e.g. to use the German expression “Java programmieren” instead of the English expression “Java programming”).

Proper names (e.g. “Harry Potter”, “Rotes Meer”) are often also difficult for taggers to recognize. A possible solution is to use a separate tool for *Named Entity Recognition (NER)*, for example SProUT [Bering et al. 2003] or the approach presented in [Klein et al. 2003]. Kouchnir [2004] uses the TreeTagger [Schmid 1994] together with the NER approach by Curran and Clark [2003] for German. Curran and Clark [2003] train a maximum entropy tagger for NER. Another solution is to directly train a tagger on a list of proper names allowed in the user input. This would be possible, for instance, for the TreeTagger. Both solutions are not viable for our approach, however, for the following reasons. Most NER tools require text corpora or a list of proper names for training. As mentioned above, we did not have training data at our disposal. We also did not want to restrict the user input to a given list of proper names, but keep it open. We therefore decided to circumnavigate proper name tagging errors with a hand-crafted heuristics. The heuristics (described in detail in Section 5.1) enables our system to recognize proper names by taking into account capitalization and German word order. It is advisable, however, to consider the integration of a NER tool again at a later stage, when enough profiling dialogs have been conducted, i.e. training material has been collected.

The TreeTagger

We decided to use the TreeTagger [Schmid 1994] for LINGUINI. It is easily accessible and comes with a chunker, which can be used for recognizing syntactic units (cf. Section 3.4.2). The TreeTagger is a stochastic tagger, which estimates transition probabilities using a decision tree. According to Schmid [1994], it achieves a higher accuracy than a standard trigram tagger. The main advantage of TreeTagger is that we can use an already trained version of it for German, which produces sufficiently correct tags for our user input. It uses the STTS tagset (cf. Section 3.3.1).

Another advantage is that TreeTagger does not only label words with part-of-speech tags but also determines the base form of words, i.e. it functions also as a *lemmatizer*. The lemmatizer function is needed in our approach for finding words in GermaNet, which only contains base forms. If the user enters inflected forms like “des Baums (of the tree)” or “die Bäume (the trees)”, these cannot be found in GermaNet, but only the base form “Baum (tree)”.

Figure 3.4 shows the TreeTagger output for the user interest “spannende Bücher lesen (reading exciting books)”. The middle column shows the part-of-speech tag for each word, the column on the right the base form determined by the tagger:

<u>word</u>	<u>part-of-speech</u>	<u>base form</u>
spannende (exciting)	ADJA	spannend
Bücher (books)	NN	Buch
lesen (reading)	VVFIN	lesen

Figure 3.4: Sample output produced by the TreeTagger

3.4 Syntactic Analysis

3.4.1 Syntactic Parsing and Grammars

In the previous section, we described how the first step in analyzing user input, part-of-speech tagging (i.e. assigning morpho-lexical categories to words), can be accomplished. The next step is to find syntactic units in our user input. The method used for this task is called (*syntactic*) *parsing*, the corresponding tools (*syntactic*) *parsers*. The term “parsing” can also refer to semantic, morphological, or text analysis [Langer and Naumann 2003]. In the following, we focus only on syntactic parsing, however. A parser checks whether a natural language expression is generated by a formal *grammar* and, in case it is, returns a structural representation of this expression, called *parse*. For the widely used context-free grammars this parse has a hierarchical structure and is hence called *parse tree*.

Existing approaches to parsing can be divided into 1) *deep parsing* and *shallow parsing* and into 2) *statistical parsing* and *rule-based* (also: *knowledge-based*) *parsing*. The first distinction refers to the depth of linguistic analysis a parser accomplishes. Some applications require the parser to perform a *deep analysis*. Here, a complete representation of the input is created which contains syntactic (and sometimes also semantic) relations between words and phrases in full detail. For this purpose, linguistically sophisticated grammar formalisms are used, such as the unification-based formalisms Head-driven Phrase Structure Grammar (HPSG) [Pollard and Sag 1994] or Lexical Functional Grammar (LFG) [Bresnan 2001]). For other applications, and also for the approach of this thesis, *partial parsing* (also called *shallow parsing* or *chunk parsing*) is sufficient. Partial parsing aims at recovering syntactic information efficiently and reliably from unrestricted text [Abney 1996a], which may often contain

“noise”, such as errors, unknown words, or unknown syntactic structure. For this reason, the *partial parser* (or: *chunker*) tries to identify reliable parts (or: *chunks*) rather than to create a complete parse tree in full detail. Hence, partial parsers create a rather flat and perhaps incomplete representation of the input. Their advantage is that they are typically more robust and faster than deep parsers. A fundamental deterministic partial parser for English and German is CASS [Abney 1996b], which is based on the technique of finite-state cascades. Another example for a finite-state based partial parser, which uses a manually constructed grammar for a subset of the German language, is described by Fliedner [2002]. An extensive overview of partial parsing is given in [Abney 1996a].

The second distinction refers to the technological basis of the parser. A *rule-based* (or: *knowledge-based*) *parser* employs a grammar consisting of a set of rules that model the syntactic phenomena of a language and were manually specified by a linguist. The drawbacks of rule-based parsing are that grammar development is very time-consuming and that syntactic input structures which are not covered by the rule set cannot be analyzed by the parser. On the other hand, rule-based parsers can operate with sophisticated grammar formalisms such as the above mentioned unification-based grammars, and for the sentences that are covered a high analysis quality can be obtained. Rule-based parsers can be distinguished from *statistical parsers* [Langer and Naumann 2003], for which probabilistic grammars with weighted rules are automatically extracted from structurally annotated text corpora. A probabilistic grammar assigns a probability to each possible analysis of a given string, hence it can be used to choose among them. In training, the rule weights are adjusted so that the corpus annotation for each sentence is predicted as the most probable analysis. If trained on a sufficiently large corpus, statistical parsers are typically very robust since it is possible to include low-weighted rules for rare phenomena and ill-formed input without causing harm in the analysis of more frequent phenomena (which relies on more heavily weighted rules).

The above described properties of statistical parsers especially help to cope with the problem of structural ambiguity. *Syntactic* or *structural ambiguity* occurs very frequently in natural language and affects all types of parsers. For a given input (e.g. “John saw the man with the telescope”), two different parse trees can be produced, which correspond to two different meanings of the sentence (cf. Figure 3.5):

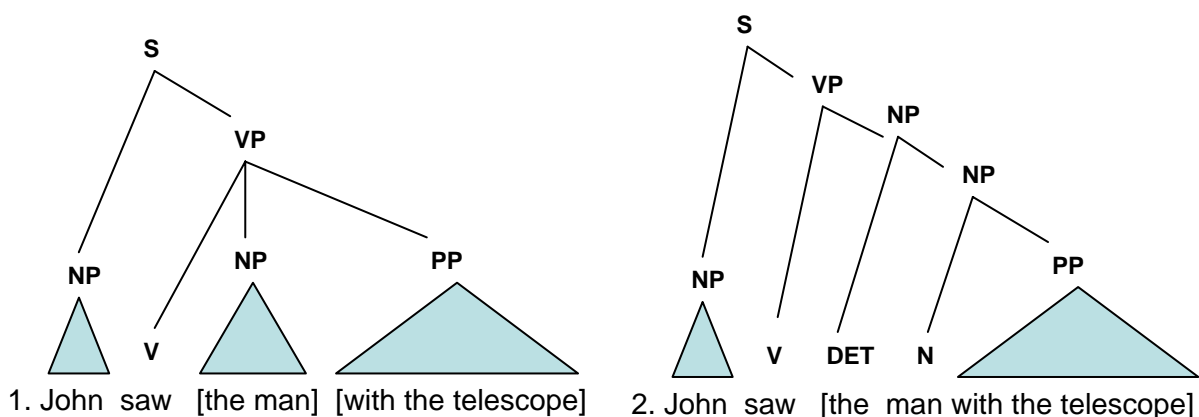


Figure 3.5 Structural ambiguity

In Figure 3.5, the left tree (1.) represents the meaning that John was in possession of a telescope with which he observed the man, whereas the right tree (2.) implies that not John but the man carried the telescope. Statistical parsers address the problem of structural ambiguity by ranking the possible parses according to their probabilities [Manning and Schütze 1999] as described above.

3.4.2 Choosing a Parser for LINGUINI

A parser is used in LINGUINI for the second analysis step: after the user input has been part-of-speech tagged in the first step, syntactic units have to be found in the second step. These syntactic units form the basis for the succeeding extraction of attributes, i.e. for building a basic semantic representation of the user interest. For finding syntactic units, we use the chunker which is associated with the TreeTagger. Training data for this chunker have been taken from the NEGRA corpus [Brants et al. 1997]. However, the chunker uses a modified form of the syntactic constituents that occur in the NEGRA corpus which is not described in a publication as yet. It finds nominal, verbal, or prepositional units, so-called *chunks*, and labels them with the following tags in angle brackets:

- <NC> spannende Bücher (exciting books) </NC> nominal chunk
- <VC> lesen (reading) </VC> verbal chunk
- <PC> im Garten (in the garden) </PC> prepositional chunk

The completely tagged and parsed user interest “spannende Bücher lesen (reading exciting books)” is shown in Figure 3.6:

```
<NC>
  spannende (exciting)/ADJA
  Bücher (books)/NN
</NC>
<VC>
  lesen (reading)/VVFIN
</VC>
```

Figure 3.6 User input, tagged and chunked with TreeTagger and its associated chunker

A chunker, which provides a shallow analysis, proved more suitable for our purposes than a deep parser (cf. Section 3.4.1) for the following reasons:

- Our user input may be “noisy”, i.e. it may contain words which are unknown to the tagger or lead to incorrect part-of-speech tags. Deep parsers are very vulnerable to these effects and might provide no result at all, whereas it is the designated task of chunkers to handle such noisy input.
- Deep parsers mostly depend on the presence of a verb for detecting syntactic relations of constituents. Our user interests are often formulated without verb (e.g. “guter Wein (good wine)”, “Computer (computers)”). Chunkers behave more robustly here, too.
- Finally, we do not need detailed knowledge about syntactic relations for extracting attributes from the user input. Detecting phrases or chunks is sufficient here.

3.5 Lexical-Semantic Networks

Lexical-semantic networks (or *wordnets*) are graphs which relate words by defining semantic and lexical relations between them. The first lexical-semantic network was the Princeton Wordnet developed for English in the early 1990s [Miller 1995]. In 1996, the EuroWordNet project began, which had the aim of developing a multilingual database with wordnets for

several European languages (Dutch, Italian, Spanish, German, French, Czech, and Estonian) (cf. www.illc.uva.nl/EuroWordNet). As a consequence, wordnet projects started for a variety of languages. One of them is *GermaNet*, a wordnet for German, which has already been integrated into EuroWordNet and is in use by a variety of (mostly research) applications. In the following, we describe GermaNet, which constitutes an important component of our profiling approach, in more detail.

3.5.1 GermaNet: A Lexical-Semantic Network for German

At present, GermaNet models the word classes of nouns, verbs, and adjectives. They are represented separately, i.e. words are only related within the same word class, but not across classes. The central object of GermaNet, which arises from the original Wordnet and is also typical for wordnets of other languages, is the *synonym set* (short: *synset*) [Miller 1995]. A synset consists of one or many members, which are called *synonyms*, *word forms*, or *lexical units*. “Sport”, “Sportart”, “Disziplin”, for instance, are different lexical units that belong to a common synset, which represents the meaning of “sports”.

As Wordnet, GermaNet semantically group words by specifying relations between them. For this purpose, there are two basic types of relations encoded in GermaNet: *lexical relations* (e.g. *synonymy* and *antonymy*), which hold between lexical units and *conceptual relations* (e.g. *hyperonymy*, *hyponymy*, *cause*, *meronymy*, *entailment*, *subevent*), which hold between synsets. A detailed account of the GermaNet-specific relations is given in [Hamp and Feldweg 1997, Kunze and Lemnitzer 2002b].

Formally, each word class is represented as a (separate) graph. Nodes in the graph are either synsets or lexical units. Edges are the lexical or conceptual relations defined between these nodes and can hence be either symmetrical or directed. The GermaNet graph structure (as opposed to a tree structure, for instance) is also motivated by the frequent use of **cross-classification**. A “cat”, for instance, is both a “vertebrate” and a “pet”. It is hence a hyponym (i.e. subordinate) node of both parent nodes, which are in turn hyponyms of a common parent node (“animal”) as is shown in Figure 3.7:

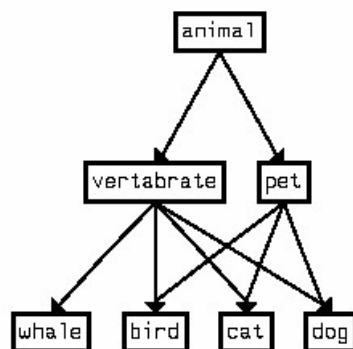


Figure 3.7: Cross-classification (source: [Hamp and Feldweg 1997])

GermaNet only implements **base forms** of words. It is assumed that external morphological analyzers map these base forms onto morphologically inflected lexical units [Hamp and Feldweg 1997]. As mentioned earlier (cf. Section 3.3.2), a lemmatizer can be used in the other direction, where a base form needs to be determined for an inflected word. This base form can then be searched for in GermaNet. Originally, **proper names** were not included in GermaNet, but some frequent and important proper names have been encoded nevertheless.

GermaNet has been built from scratch, i.e. it is neither a translation of the English Princeton WordNet, nor is it based on a single dictionary as source [Hamp and Feldweg 1997]. This

process has been supported by lemmatized frequency lists extracted from corpora, the CELEX German lexical database, and the automatic extraction of selectional restrictions of verbs. Differences between WordNet and GermaNet are described in detail by Hamp and Feldweg [1997] and Kunze and Lemnitzer [2002a]. GermaNet has not yet reached the coverage of WordNet, which is also due to its later start, but is growing constantly. At the beginning of 2005, GermaNet has covered about 42 000 synsets with about 62 000 lexical units [Kunze and Lemnitzer 2005], whereas WordNet currently covers more than 117 000 synsets (source: <http://wordnet.princeton.edu/man/wnstats.7WN>).

Wordnets can serve various purposes, for instance, as a training source for statistical methods in NLP [Hamp and Feldweg 1997]. Recent applications of GermaNet are, for example, the management of product requirements [Jörg 2003] or an improved search method for the Internet [Kruse et al. 2005]. Wordnets like GermaNet can also be used for calculating the semantic relatedness between words [Gurevych and Niederlich 2005a]. Most efforts, however, are still directed towards further improving the resource GermaNet (e.g. by adding textual conceptual definitions (“glosses”) [Gurevych and Niederlich 2005a]), relating it to other linguistic resources, and finding methods for its evaluation and automatic acquisition. An interesting report about practical experiences gathered during the application of GermaNet for analyzing documents of specific domains has been published by Kunze and Rösner [2003]. One of its outcomes is a “GermaNet user’s wish list” for the future. However, there do not seem to exist any profiling approaches which make use of GermaNet.

3.5.2 Using GermaNet for Profile Construction and Update in LINGUINI

In our approach, the user model is based on an extended representation of GermaNet. We only use a subset of the GermaNet relations, however. In the following, we explain our specific view on this resource in detail. We interpret GermaNet as a **directed acyclic graph**, which consists of **synset nodes**. **Edges** of the graph are either **hyponym relations** (i.e. the relation “x is *hyponym* of y” is directed from child node x to parent node y) or **hyperonym relations** (i.e. the relation “x is *hyperonym* of y” is directed from parent x to child y). “Wassersport (water sports)”, for instance, is a hyponym of “Sport/Sportart/Disziplin (sports)” and “Sport/Sportart/Disziplin (sports)” is a hyperonym of “Wassersport (water sports)”.

Each user profile instance in our approach is a subgraph of the GermaNet graph, which is extended by **semantically structured nodes** representing complex user interests (cf. Section 4.4.2.1). Not all GermaNet synsets are contained in a user profile, only the synsets that are relevant for modeling the interests of a particular user.

We decided to use GermaNet, because it supports our goals in two important respects:

- **Dialog control:** By using GermaNet as the basis for our user model, we are able to obtain information about the semantic relatedness of information items stored in the graph. This information is used for deciding on the next dialog step.
- **Handling not predefined user information:** GermaNet moreover serves as a knowledge base for inserting new, not predefined user information at the semantically correct position in the profile graph.

Another advantage is that a wordnet allows the user model to contain synsets instead of single lexical units (cf. [Semeraro et al. 2005]). In our case, the name of each user interest is represented as a synset comprising several synonym word forms, if available. A profile which stores synsets is presumably of better use for recommending products than a profile without synonyms. If the profile, for example, contains the synset {Sport, Sportart, Disziplin} (all representing the meaning “sports”), the recommender system can find items in the product

database that are labeled with any of the three lexical units. If the profile only contained the lexical unit “Sport” (not the entire synset), items labeled with “Sportart” or “Disziplin” could not be found in the database.

We also had to solve some problems with respect to GermaNet. The first problem was that there are no off-the-shelf tools, as for instance a JAVA-API, for accessing GermaNet’s knowledge [Gurevych and Niederlich 2005a]. The second problem was the already mentioned smaller coverage of GermaNet with respect to synsets and lexical units. As a consequence, some words entered by users cannot be found in GermaNet. This was, for instance, noticeable with respect to nominalizations (“das Wandern (hiking, as a noun)”). These problems mainly affected the implementation of our prototype system. The problems, together with our solutions, are therefore described in detail in Chapter 6.

3.6 Word Sense Disambiguation

3.6.1 The Problem of Ambiguity

In natural languages, individual words or larger syntactic units can be *ambiguous*, i.e. they can have several possible interpretations. Since ambiguity is a very frequent problem, it needs to be addressed for making NLP systems truly viable in practice. In Section 3.3.1, we have already introduced *word class ambiguity* with respect to tagging and, in Section 3.4.1, we have shown an example of *syntactic or structural ambiguity* with respect to syntactic analysis. Besides these, the term *lexical ambiguity* refers to words which have several meanings or senses, such as “Dichtung” shown in Example 3.1:

Example 3.1

Dichtung

Sense 1: poetry – a literary form

Sense 2: sealing – a rubber object used to prevent the discharge of liquids

Larger syntactic units (e.g. sentences) are often also affected by ambiguity which is, for example, introduced by *pronouns* (it, him, she, they, etc.), as illustrated in Example 3.2 (source: [Brockmann 2002]):

Example 3.2

Peter mag seinen Hund, obwohl er ihn manchmal beißt.

Sense 1: Peter likes his dog, although it bites him sometimes.

Sense 2: Peter likes his dog, although he bites it sometimes.

3.6.2 Existing Approaches to Word Sense Disambiguation

In the solution presented in this thesis, word class ambiguity and structural ambiguity can sufficiently be handled by the TreeTagger and its associated chunker. More difficult is the problem of lexical ambiguity, on which we concentrate here. Manning and Schütze [1999] define *word sense disambiguation (WSD)* as the task “to determine which of the senses of an ambiguous word is invoked in a particular use of the word”. The following, more complex definition is given by Ide and Véronis [1998]:

Definition 3.1 (Word sense disambiguation (WSD) [Ide and Véronis 1998]): Word sense disambiguation involves the association of a given word in a text or discourse with a definition or meaning (sense) which is distinguishable from other meanings potentially attributable to that word. The task therefore necessarily involves two steps:

- (1) the determination of all the different senses for every word relevant (at least) to the text or discourse under consideration; and
- (2) a means to assign each occurrence of a word to the appropriate sense.

We exemplify these two steps in the following. In order to **determine all different senses of relevant words (Step 1)**, many approaches use predefined senses listed in dictionaries or synonyms listed in a thesaurus. Another possibility is to extract word senses from lexical-semantic networks, such as WordNet or GermaNet. The definition of the term “sense”, however, is a difficult problem, since a sense is often not completely clear-cut, as pointed out by Manning and Schütze [1999], but can either be a subsense of some other sense or differences in meaning between two senses can be fuzzy.

The aim of **Step 2** is to determine **which of the theoretically possible senses should be assigned to a specific word occurrence**. Ide and Véronis [1998] mention two major sources of information to determine the sense. The first source is *context*, which includes information contained in the text or discourse, in which the word appears, and also extra-linguistic information or meta information about the text, for example, about the situation of the author. Secondly, *external knowledge sources* (e.g. lexicons or encyclopedias) can be used for disambiguation. On the basis of this distinction, existing approaches to WSD can be divided into corpus-based (or: stochastic) and knowledge-based (or: non-stochastic) methods (e.g. [Ide and Véronis 1998, Montoyo et al. 2005]).

The first type, *corpus-based* or *stochastic approaches*, uses information about the contexts of previously disambiguated word instances derived from corpora [Ide and Véronis 1998]. Machine learning algorithms, such as supervised learning and unsupervised learning play a crucial role here (cf. [Manning and Schütze 1999]). While *supervised learning* can be seen as a classification task and is based on training sets, in which each piece of data is labeled with a sense, *unsupervised learning* can rather be considered a clustering task and is based on unlabeled text. Another possibility is *dictionary-based disambiguation*, where sense definitions that occur in lexicons are exploited directly (e.g. [Lesk 1986]). Disadvantages of stochastic approaches are, for example, the effort to create labeled training corpora for supervised learning and the sparse-data problem. The *sparse-data problem* refers to the fact that even for large sets of training data, it cannot be guaranteed that all possible senses of a word occur in the corpora in their actual frequency.

The second type, *knowledge-based* or *non-stochastic approaches*, relies on extensive external knowledge sources. These approaches often employ technologies from artificial intelligence, logic formalisms, ontologies, and lexical-semantic networks. A detailed account of these systems is given in [Ide and Véronis 1998]. Many authors (e.g. [McRoy 1992, Montoyo et al. 2005]) argue that WSD requires *hybrid approaches*, combining several knowledge sources (e.g. syntagmatic, paradigmatic, or statistical information). Although WSD has been a topic for systems dealing with natural language since the 1950s, no large-scale, broad-coverage WSD system has been built yet [Montoyo et al. 2005]. While early approaches to WSD were mainly knowledge-based, most of today’s systems follow a stochastic approach [Navigli and Velardi 2005]. WSD is applied, for example, in the areas of information retrieval and Internet search machines, question answering, information extraction, machine translation, grammatical analysis, and speech processing. In the following section, we describe which specific aspects of WSD we have to address in our approach and present some existing approaches that are particularly relevant for our work.

3.6.3 Word Sense Disambiguation in LINGUINI

Ambiguity in User Interests

During the profiling dialog, ambiguities may occur in the interests entered by users. If a user enters, for instance, one of the three phrases about “Dichtung” shown in Example 3.3, LINGUINI has to decide whether the sense of poetry or sealing is meant. Both senses (poetry and sealing) are theoretically possible in all of the three cases (although the poetry sense might intuitively be more likely):

Example 3.3

Dichtung aus China	(poetry/sealing from China)
schöne Dichtung	(beautiful poetry/sealing)
besonders moderne Dichtung	(particularly modern poetry/sealing)

As described in the previous section, the intended meaning of a word can often be inferred from its context. In some cases, co-occurring words with clear meaning (cf. (1) in Example 3.4) or morphological properties of words (cf. (2)) may help for disambiguation:

Example 3.4

- (1) moderne Dichtung im Literaturhaus München
disambiguation by means of the additional word “Literaturhaus (location for literary events)”
- (2) moderne Dichtungen
the plural ending “Dichtung-en” is only possible for the sense of sealing, but not for the sense of poetry

A problem with respect to our user input is that context is rather sparse. Interests are either formulated as isolated words (e.g. “Dichtung”) or embedded in rather short phrases (e.g. “besonders moderne Dichtung”), in which helpful co-occurring words scarcely occur. In order to make use of disambiguating morphological properties of words, such as word endings or plural forms (e.g. “Bänke (benches)” vs. “Banken (banks)”), linguistic knowledge as encoded in lexicons would be necessary.

Even if several interests are entered together (cf. Example 3.5), their meanings are often completely different and they may not serve as a reliable contextual source for disambiguation:

Example 3.5

Dichtung; guter Wein; wandern mit der Familie; ins Kino gehen; Hunde
(poetry/sealing; good wine; hiking with the family; going to the cinema; dogs)

A possible solution is to ask the user for clarification in cases where the system cannot resolve the ambiguity automatically or no disambiguation component exists at all, such as suggested by [Felfernig et al. 2004]. This is problematic, however, since the user must not be bothered with too many additional questions, as outlined in Chapter 2.

A Disambiguation Solution for LINGUINI

Since we do not have the necessary amounts of training data at our disposal, we cannot apply the corpus-based approaches mentioned above. The fact that we employ GermaNet for profile and dialog management already makes knowledge-based solutions that are related to lexical-

semantic networks most interesting for us. In the following, we briefly discuss existing approaches of this type.

A possible solution is to apply a dictionary-based approach, such as the one developed by Lesk [1986], without dictionaries but together with Wordnet. This is possible by using a specific type of information contained in WordNet, the so-called “glosses” (i.e. textual conceptual definitions) instead of dictionary definitions. Unfortunately, GermaNet does not contain any glosses. Gurevych and Niederlich [2005a] compensate this drawback by generating artificial textual definitions of synsets (“pseudo-glosses”) automatically by exploiting relations encoded in GermaNet.

Another possibility is to use lexical-semantic networks to calculate semantic relatedness between synsets and use this information for disambiguation. Budanitsky and Hirst [2001] explain that *semantic relatedness* is a more general term than *semantic similarity*. Semantic non-similar words (e.g. “hot” – “cold”) may be semantically related nevertheless by means of lexical or conceptual relations within the lexical-semantic network as introduced in Section 3.5.1. Several evaluations have been conducted which compare existing algorithms for calculating semantic relatedness for English and German (e.g. [Budanitsky and Hirst 2001, Gurevych and Niederlich 2005a]). All of them confirm that, for both languages, the algorithm developed by Jiang and Conrath [1997] performs best in measuring the semantic relatedness of words.

Most existing methods for calculating semantic relatedness use WordNet for English and operate on pairs of *nouns*, i.e. they do not take into account verbs. *Verbs*, however, have special semantic properties – so-called *selectional preferences* (e.g. [Brockmann 2002]) – which may cause contextual dependencies between words as shown in Example 3.6. Hence, verbs also play an important role for disambiguation.

Example 3.6

The verb “essen (to eat)”, for example, imposes the semantic restriction on its subject that it has to be a living creature and on its object that it has to be edible.

The approach presented by Brockmann [2002] automatically predicts selectional preferences, i.e. which other words are most likely to occur in the context of a given verb. It also requires extensive text corpora, however. A method for calculating semantic similarity between verbs on the basis of a lexical-semantic network is proposed by Resnik and Diab [2000]. This approach counts the number of edges of the shortest path between two verb synsets: the smaller the number of edges, the more similar the two synsets. The path is given by means of the hyperonym relation (cf. Section 3.5.2).

On the background of these considerations, we propose a **hybrid disambiguation strategy** for LINGUINI. This strategy combines three methods. First, a modified version of the above mentioned algorithm developed by Jiang and Conrath [1997] can be employed. A second method can be applied later on, when the system has been in use for a while and a set of different user profiles has been acquired. This second method takes into account contextual information stored in other user profiles. Finally, the user can also be asked for clarification in case the system cannot resolve the ambiguity automatically and the “strength” of this ambiguity exceeds a certain threshold. In the following, we explain this idea in more detail.

Modified Jiang and Conrath algorithm: Originally, the algorithm by Jiang and Conrath [1997] combines a lexical taxonomy structure with corpus statistical information about the distribution of words. This algorithm can be applied in a modified form, however, relying merely on the taxonomy, without the necessity of text corpora [Seco et al. 2004]. Seco et al. [2004] show that the *information content (IC)* of a word can be determined by exploiting the hierarchical structure of WordNet. They assume that synsets which contain hyponyms (subordinate nodes) contain less information than synsets which are leaf nodes (i.e. most specific

nodes) in the taxonomy. Hence, they express the IC of a synset in WordNet as a function of its hyponyms:

$$IC(s) = \frac{\log \frac{\log hypo(s) + 1}{max_{wn}}}{\log \frac{1}{max_{wn}}} = 1 - \frac{\log hypo(s) + 1}{\log max_{wn}}$$

This formula calculates the *IC* of a synset *s*. The function *hypo* thereby returns the number of hyponyms of *s* and *max_{wn}* is a constant, which is set to the maximum number of synsets contained in the taxonomy. The denominator is equivalent to the most informative synset. It serves as a normalizing factor by assuring that the result is contained in [0,...,1]. This formula guarantees that the IC decreases monotonically when traversing the taxonomy from leaf nodes to root node [Seco et al. 2004]. On this basis, the semantic distance *dist* of two synsets *s*₁ and *s*₂ can be calculated by means of the following formula, with *lcs_{s1,s2}* being the lowest common subsumer of *s*₁ and *s*₂ [Jiang and Conrath 1997]:

$$dist(s_1, s_2) = IC(s_1) + IC(s_2) - 2 * IC(lcs_{s_1, s_2})$$

Finally, the semantic similarity *sim* between the two synsets can be calculated on the basis of the distance (for further details cf. Eiseler [2005]):

$$sim(s_1, s_2) = 1 - dist(s_1, s_2)$$

Contextual information contained in other profiles: The second part of the solution is related to the fact that topics (e.g. sports activities, reading habits) and also specific formulations of interests (e.g. “mit meinen Kindern spielen (playing with my kids)”) may reoccur in different profile instances. As opposed to the approach suggested by Gurevych and Niederlich [2005a], the semantic relatedness of interests (synsets) will not be determined on the basis of comparing their glosses, however. Instead, an interest entered by a specific user is compared with *attribute values* of interests stored in other user profiles. A modified algorithm of Lesk can be employed for this task (cf. [Eiseler 2005]). This strategy is more reliable, the larger the profile data set. For a large data set, frequencies can be taken into account, which indicate the probability of a specific meaning (cf. Example 3.7):

Example 3.7

If 97 users have entered “Dichtung” together with literary contextual information (e.g. “Dichtung von Goethe (poetry by Goethe)”, “Dichtung lesen (reading poetry)”) and only 3 users have entered Dichtung in the sense of sealing then this information can be used to decide that the user under consideration has probably also meant the literary sense.

Asking the user for help during the disambiguation process should also be considered, as a fallback solution. For regulating such clarification interactions with the user, we suggest distinguishing *weak ambiguities*, for words which differ only a little in meaning, and *strong ambiguities*, for those differing considerably. The degree or strength of an ambiguity can be measured by means of the number of edges between these synsets in GermaNet (cf. [Resnik and Diab 2000]): the shortest path between two weakly ambiguous words consists of fewer edges (i.e. they are closer to each other in the graph) than the shortest path between two

strongly ambiguous words (i.e. they are further away from each other). An empirically defined *confidence threshold* may help to distinguish between the two cases. If the confidence value of a word sense determined by the system (by employing the modified Jiang and Conrath or modified Lesk algorithm) passes this threshold, the user is not asked for confirmation. If the confidence value does not reach the threshold, i.e. the system is undecided, the user is asked which sense she has originally intended. It has to be investigated yet, how the choices can best be presented to the user and how this interaction could be modeled. Hence, the user is only actively involved in the resolution of strong ambiguities and unnecessary clarification interactions can be avoided.

The disambiguation process can be embedded in the LINGUINI profiling process immediately after a user answer has been received and before the information is stored in the profile. A first version of this solution has been implemented in a diploma thesis [Eiseler 2005]. The implementation is based on a previously used GermaNet-API [Ast 2003], however, which proved not fast enough for practical use (cf. Chapter 6.2.3.2). Meanwhile, the LINGUINI prototype uses a faster API [Gurevych and Niederlich 2005b], which has a different functionality. The disambiguation module could therefore not be evaluated in connection with the final user study. For future use, the disambiguation module has to be adapted to the functionality of the currently used GermaNet-API. It has to be emphasized that word sense disambiguation was not the main focus of this thesis and has not been investigated in complete detail here.

3.7 Speech Dialog Systems

Speech Dialog Systems enable the communication between user and system in spoken natural language. For this purpose, they need components for speech recognition, language understanding, dialog management, communication with external systems (e.g. databases), response generation, and speech synthesis [McTear 2002]. Most dialog systems are designed for **speech**, although several approaches (e.g. Bainhat, www.brainhat.com or Galaxy Communicator, communicator.sourceforge.net) are also adaptable for **typed input**, as is common in E-Commerce platforms for example. In the following, we refer to both system types with the term *dialog system*. If we refer to special properties of either speech or text dialog systems, we will state this explicitly. First, we focus on strategies for dialog management. Afterwards we discuss in-vehicle use of dialog systems in more detail, before we describe the dialog strategy chosen for LINGUINI.

The topic of evaluating dialog systems is addressed in more detail in connection with the evaluation of LINGUINI in Chapter 6. For an in-depth discussion of the characteristics of human language (e.g. turn-taking, speech acts, etc.) as well as possible dialog system architectures we refer to Jurafsky and Martin [2000].

3.7.1 Dialog Management

The profiling dialog is controlled by a dialog manager. The dialog manager determines the reaction to a given user input. Possible reactions during a profiling dialog may be, for instance, selecting a suitable next question, updating the user profile, or terminating the dialog if certain conditions are met. In this section, we first discuss possible dialog control strategies and then look at different types of knowledge sources which can be used for controlling the dialog flow.

Dialog Control Strategies

[McTear 2002] classifies dialog systems according to the control strategies they use into finite state-based, frame-based, and agent-based systems.

Finite State-Based Systems

A *finite state-based* (also called *graph-based*) approach is used by most available commercial dialog systems [McTear 2002]. The dialog in today's commercial systems is therefore still mostly predetermined and the user has no influence on dialog control [Carstensen et al. 2004]. These systems typically use a sequence of predetermined states through which the user is led. The dialog flow is modeled as a set of dialog states. Transitions between states denote possible paths through the graph. At each state, the user is prompted with a question. The system then reacts to the user answer.

An advantage of finite state-based systems is that their implementation is simple. They mainly restrict user input to single words or phrases, which permits vocabulary and grammar for each state to be predefined. This strategy is also less demanding for the speech recognition module and makes the overall system very robust. The drawbacks are, however, that the dialog flow is rather inflexible and is often perceived as not very natural. Moreover, by being restricted to answer with predefined words the user has very little freedom of expression. Corrections are difficult or impossible in these systems and users in general cannot take the initiative, ask questions, or introduce new topics. Dialogs where a negotiation between user and system is required cannot be modeled with this approach. A solution would be to augment finite state-based systems with a natural language understanding component that accepts more freely formulated input in the form of more complex phrases or partial sentences. Finite state-based systems are most suitable for very well-structured tasks the subtasks of which are easy to predict. Tasks suitable for finite state-based systems include, for instance, providing timetable information, selling train or flight tickets, or booking hotels. A sample finite state-based system for an automatic book club service is described by Larsen and Baeekgaard [1994]. Another example is the Nuance automatic banking system described in [McTear 2002], which is able to conduct a dialog as is shown in Figure 3.8:

System: *What would you like to do?*
User: *Pay a bill.*
System: *What company would you like to pay?*
User: *Midland Bank.*
System: *How much would you like to pay?*
User: *One hundred and twenty-five pounds fifty-seven pence.*
System: *What date would you like the payment to be made on?*
User: *Monday.*
System: *You want to pay HSBC one hundred fifty-five pounds and fifty-seven pence on April 5th? Is this correct?*
User: *Yes.*
System: *Your payment will be sent. Would you like to pay another bill?*

Figure 3.8: Sample dialog with the Nuance automatic banking system (source: [McTear 2002])

Frame-Based Systems

Frame-based systems typically accomplish a form-filling task, for which a certain amount of information needs to be collected [McTear 2002]. For these systems, the underlying dialog control structure is not a graph, but a *frame* or *template*, which has slots to fill. This frame serves as a dialog model which keeps track of information yet to be acquired. The slots represent information the system has to elicit (e.g. departure time or destination in a train booking system) and may be filled in any given order. Questions are predefined and can either be executed in a particular order or may possess preconditions, which have to be true for the questions to be asked.

Different variations of frames can be found. VoiceXML, for instance, uses *forms*, which consist of field and control items. A field is used for collecting information, whereas a control item specifies procedural statements for prompting and computation [McTear 2002]. Another variant are *E-forms* (e.g. [Goddeau et al. 1996]), where slots can have different priorities for different users and information in slots can be related. Further data structures, such as *schemas* or *task structure graphs*, can be found in [McTear 2002].

System1:	<i>Good morning. This is the automatic timetable information service. How can I help you?</i>	System6:	<i>At what time would you like to go from Munich to Hamburg on Sunday the 20th of November?</i>
User1:	<i>I need a connection from Hamburg to Munich.</i>	User6:	<i>I have to arrive in Hamburg late in the evening.</i>
System2:	<i>When would you like to go from Hamburg to Munich?</i>	System7:	<i>So you want to arrive in Hamburg between nine p.m. and eleven p.m.?</i>
User2:	<i>Next Friday at eight o'clock.</i>	User7:	<i>Yes.</i>
System3:	<i>So you want to travel on Sunday the 20th of November at eight o'clock a.m.?</i>	System8:	<i>There are three connections. This is the first one: With IntercityExpress 880, departure from Munich at 15:54, arrival in Hamburg at 21:58. Do you want the next connection?</i>
User3:	<i>No, on Friday.</i>	User8:	<i>No, thanks, that one is fine.</i>
System4:	<i>So you want to travel on Friday the 18th of November at eight o'clock a.m.?</i>	System9:	<i>Do you need yet another connection?</i>
User4:	<i>Yes.</i>	User9:	<i>No, thank you.</i>
System5:	<i>There is the following connection: With IntercityExpress 793, departure from Hamburg at 8:02, arrival in Munich at 14:05. Do you want me to repeat the connection?</i>	System10:	<i>We wish you a pleasant journey. Thank you for calling. Good bye.</i>
User5:	<i>No, but I need a train for the return trip on Sunday.</i>		

Figure 3.9: Sample dialog with the Philips Train Timetable Information System (source: [McTear 2002], original German source: [Aust et al. 1995])

Frame-based systems are normally flexible enough to handle additional information given by a user and can also recognize several information items which occur in a single user input. Therefore, they enable information to be acquired in less time than with a finite state-based system. Normally, frame-based systems keep the control, but they can support user initiative to a limited degree. These systems are more user-friendly in that the dialog is event-driven and not predetermined. However, the context frame-based systems take into account for determining their next action is mainly restricted to the user's last utterance, the state of the frame, and some priorities for dialog control. More complex transactions cannot be modeled. Moreover, it is difficult for the developer to determine in which context and under which preconditions a rule has to fire, i.e. a question should be asked. This difficulty increases with the

size of the rule base. Examples for frame-based systems are the Philips Train Timetable Information System [Aust et al. 1995], Philips SpeechManiaTM, or the Communicator system described by Ward and Pellom [1999]. Figure 3.9 above shows a sample dialog conducted with the Philips Train Timetable Information System.

Agent-Based Systems

From all dialog system types discussed here, *agent-based systems* have obtained the most attention in the scientific community in recent years. They allow a more complex communication by employing dialog strategies that originate from the field of artificial intelligence. A communication is modeled as interaction between intelligent agents, in most cases allowing a mixed-initiative dialog, where the user can control the dialog and make contributions which are not constrained by the previous system utterance. There are a variety of different agent-based dialog systems depending on particular aspects of intelligent behavior included in the system (cf. [McTear 2002]). Some systems use *theorem proving* (e.g. Circuit-Fix-It Shop [Smith and Hipp 1994]). In *plan-based approaches*, utterances are treated like actions in a planning system, which are performed to achieve some goal (e.g. TRIPS [Allen et al. 2001]). Further approaches are *conversational agency* (e.g. TRAINS [Ferguson et al. 1996]), *event-driven dialog in a distributed architecture* (e.g. SUNDIAL [McGlashan et al. 1990]), and *dialog as rational interaction* (e.g. ARTEMIS [Sadek et al. 1997]).

The advantage of agent-based systems is that they enable a more natural, more complex dialog interaction than is possible with finite state-based and frame-based approaches. They are able to solve problems collaboratively with the user. Their mixed-initiative control strategy gives more freedom to the user. Here, both system and user can ask questions, correct previous utterances, or change the topic. An obvious disadvantage is that they are rather difficult to develop. The dialog agent has to maintain a task model and dialog states and has to reason over them. Moreover, the beliefs and intentions of both agents are involved [McTear 2002]. These systems also require more sophisticated NLP, involving a deeper semantic processing, as opposed to the pattern-matching and concept-spotting techniques often used in finite state and frame-based systems.

System1: *Is there a wire between connector 84 and connector 99?*
User1: *It is not there.*
System2: *Add a wire between connector 84 and connector 99.*
User2: *I need help.*
System3: *Obtain a wire.*
User3: *Done.*
System4: *Connect metal of one free end of the wire to connector 84.*
User4: *How?*
System5: *Bend back connector 84.*
User5: *Done.*
System6: *Insert free end of wire into connector 84.*
User6: *Wire between 84 and 99 is connecting.*
System7: *What is the LED displaying?*

Figure 3.10: Cutout of a dialog with Circuit-Fix-It shop (source: [McTear 2002])

Figure 3.10 shows a sample dialog with Circuit-Fix-It Shop, which helps users to fix an electronic circuit. Both parties, user and system, need to exchange knowledge and facts about the task in order to solve it. The system also adapts to the user's level of experience by providing more detailed information if required.

Knowledge Sources for Dialog Management

A dialog manager may use several distinct knowledge sources. Sometimes, these are referred to collectively as the *dialog model* [McTear 2002]. Such knowledge sources are, for instance:

- **Dialog history:** a record of the dialog so far, used to support NLP
- **Task record (also: form, template, or status graph):** represents the information that needs to be collected
- **World knowledge model:** supports common sense reasoning of the system
- **Domain model:** represents information relevant for a specific domain
- **Generic model of conversational competence:** contains information about turn-taking, discourse obligations, etc.
- **User model:** contains information about the user which may be relevant for the dialog

The use of these knowledge sources depends on the dialog strategy. In *finite state-based systems*, these models are often not used or only exist implicitly in the system in a simplified form. Here, a user model (if present at all), for example, contains very little information which can directly be used to control the dialog flow. In *frame-based systems*, an explicit task record is used to control the dialog flow. User models, in case they exist, are equally simple, as in finite state-based systems. This is different for *agent-based systems*, which often require a complex user model and a dialog history to determine how the system should react. They often also make use of discourse structure modeling and represent intentions, goals, and beliefs of the user.

3.7.2 In-Vehicle Application of Dialog Systems

The application of dialog systems for user profiling and adaptive systems has already been discussed in Chapter 2. Here, we will therefore concentrate on dialog systems for in-vehicle use. In-vehicle application (as well as telephone application) justifies the use of speech dialog systems more than any other application scenario. In the first place, this is due to the fact that speech – as opposed to typed interaction – significantly improves driving safety (cf. [Gärtner et al. 2001, Bühler et al. 2003]).

Although during the last decade a lot of research activities have been conducted by major car manufacturers (also as joint projects with universities), and many high-end cars integrate speech dialog facilities, robust in-vehicle dialog still seems to be a difficult task. One reason is that background noise complicates speech recognition in the vehicle. Poor speech recognition may in turn cause problems for the other components of language understanding. For instance, word-level ambiguities may increase considerably. Moreover, the main concentration of the driver is on the traffic rather than on the dialog, which sometimes may cause an interruption or delay in speech interaction. The dialog has to be flexible enough to adapt to these situations and also has to allow for corrections, in case the driver was distracted and wants to modify or correct an utterance she has made. This gives an impression of the difficulties and explains why most of these systems are still in a developmental phase. In the following we discuss some examples of recent approaches.

In the EU project SENECa (Speech control modules for Entertainment, Navigation, and communication Equipment in Cars) (cf. [Gärtner 2001]) a speech dialog system has been de-

veloped which also allows for multimodal interaction. The system enables access to a variety of entertainment, navigation, and communication applications [Minker et al. 2003]. In SE-NECa, the above mentioned problem of low speech recognition confidence and word-level ambiguity is addressed by conducting clarification dialogs with the user. As a fallback solution, in case of misrecognition, the user can also spell the word.

Another project investigating multimodal human-machine interaction in mobile and other scenarios is SmartKom [Wahlster et al. 2001]. SmartKom is the follow-up project of VERBMOBIL [Wahlster 2000], at its time (1993 – 2000) the worldwide largest project conducted on dialog systems. Under the leadership of the German Research Center for Artificial Intelligence (DFKI), SmartKom partners are various universities, private research institutions, and industry (DaimlerChrysler, Philips, Siemens, and Sony). SmartKom aims at developing intuitive user interfaces which integrate the recognition of speech, gesture, and facial expressions as well as graphical display of text, maps, and animations. The concepts developed in SmartKom have been tested in different application scenarios described by Bühler et al. [2002]. Most relevant for this thesis is the “SmartKom Mobile” scenario, where the SmartKom Mobile device can either be carried by a pedestrian or can be installed in a car. In the latter case, it takes into account car-specific functionality such as real-time information about the car state. The salient task of SmartKom Mobile in the car scenario is the navigation functionality, which involves a trip planning phase and a trip execution phase. During the *trip planning phase*, the user can specify certain properties of the trip or get information about trip-related aspects, for instance points of interest nearby. During the *trip execution phase*, the system supports the driver with timed presentation of driving directions and processes incoming positioning information. The system had to be adapted to certain restrictions imposed by the car-scenario, for instance, gestures of drivers are prohibitive whilst driving and recognizing facial expressions is difficult because drivers may frequently move their heads. For this reason, five interaction modes are assumed, in which the interaction is adapted to different driving situations (cf. [Bühler et al. 2002]). The system does not yet include user profiling. An important observation for this thesis is, however, that Bühler et al. [2002] point out that their approach would clearly profit from being adaptable to user preferences and to the user’s experience with the system.

The EU project TALK (Tools for Ambient Linguistic Knowledge, www.talk-project.org), which started in 2004, also investigates in-vehicle dialog. Partners are several universities (Saarbrücken, Edinburgh, Sevilla, Cambridge, Göteborg), DFKI, Linguamatics, BMW, and Bosch. The aim of TALK is to develop flexible adaptive, multimodal dialog systems for interacting naturally with devices and services in the car or at home. The fundamental concept for the dialog approach is the *Information State Update (ISU)* (e.g. [Larsson and Traum 2000]), which allows storing all information about the course of the dialog in the *information state* of the system. This information state is then used to calculate an appropriate reaction. An advantage of ISU is that it supports a much richer context representation than is possible in the most commonly used approaches in industry, such as finite state-based systems and VoiceXML. The project extends the ISU approach for adaptive multimodal and multilingual interaction. In the dialog approach taken in TALK, user-adaptiveness, robustness, and reusability of the dialog approach play a major role.

An in-vehicle system which recommends routes from a source address to a destination address by taking into account user preferences is the Adaptive Road Advisor [Rogers 1999]. The system presents several options to the driver, who can request changes or state additional preferences. During the dialog, a satisfactory solution is worked out between driver and system. The user model is acquired implicitly by inducing preferences from the driver’s route choices. The user model may be validated or corrected during the dialog interaction.

Hataoka et al. [2004] present a noise-robust speech recognition system for telematic systems (i.e. car information systems) and the dialog manager CAMMIA (Conversational Agent

for Multimedia Mobile Information Access) based on VoiceXML. CAMMIA extends the VoiceXML approach by using a more comprehensive NLP, which also makes use of lexicons and grammars, in order to make dialog interaction more flexible.

It can be summarized that besides multimodality, adaptiveness to user preferences and user experience is one of the most important topics for in-vehicle dialog systems. With this thesis we hope to contribute to the improvement of user modeling approaches for mobile use.

3.7.3 Choosing a Dialog Strategy for LINGUINI

For most of the above mentioned in-vehicle dialog systems, the central task is not yet to acquire user profiles and to provide adaptive services. They first have to tackle major problems such as robustness in speech recognition, ambiguity, and adaptability to the driving situation. For almost all of these systems, however, adapting the dialog to user preferences is a desirable future goal.

In Chapter 1, we have described the main characteristic properties of an adaptive in-vehicle profiling dialog system for user interests (cf. Section 1.2) and we have explained why we decided to build most of our system, including the Dialog Manager, from scratch (cf. Section 1.3). One of the main reasons for not using an existing dialog system was that none could be found that already integrates the following requirements or allows for an easy extension in this respect:

- The dialog system needs to be extensible for a **semantic knowledge representation** of user information (e.g. based on GermaNet), which allows us to find related interests. It also needs to be able to integrate a **target group model representing user interests**.
- The Dialog Manager needs to **closely interact with the semantic knowledge representation and the target group model** for storing, accessing, and comparing user interests. It has to be possible that both semantic knowledge representation and target group model influence the question selection strategy of the Dialog Manager.
- The dialog design and strategy have to enable the **acquisition of personal interests**. In particular, the preferred user input, as shown in our first user study (cf. Section 6.1), needs to be processable, and questions need to be closely tailored to this input. In particular, the system needs to **handle free, previously not defined, user input**.

Most of the above mentioned approaches are tailored for a specific application, i.e. their grammars, rule bases, vocabulary, and dialog strategy are defined with respect to the application to a specific domain or task. Moreover, many of them integrate more complex dialog models and processes – often based on linguistic theories – than are required for the profiling dialog we have in mind. Adapting them for our purposes would have required at least as much time and effort as building a new system. Our main aim was to show the general feasibility of an adaptive profiling dialog for personal interests. A question for future investigation, however, might be whether further dialog functionalities, such as a mixed-initiative or negotiation strategy, are able to increase user satisfaction or profile quality.

The dialog approach taken in LINGUINI is characterized by the following aspects. LINGUINI **allows free user input** not predefined in the system and employs a **flexible and adaptive questioning strategy**. The system adapts to topics the user has mentioned and finds new, semantically related topics dynamically by means of GermaNet and the Target Group Model. We use preconditions according to which questions are selected. Besides this, LIN-

GUINI also offers some **predefined answer possibilities with respect to details (attributes)** of a particular user interest. This is because we want to keep a common attribute structure for all interests stored in order to make them reusable. Recommender systems can then adapt their filtering methods to make use of this attribute information consistently throughout the profile. Furthermore, we decided to **leave dialog initiative up to the system**, because of the experience often reported for E-Commerce and call center applications that most customers prefer to be led by the system. According to the report of a representative of VoiceObjects (www.voiceobjects.com) given during a panel discussion at the GLDV symposium 2005, 90 % of the callers of a call center hang up the phone if a computer system asks the question “Hello, what can we do for you?”. The problem is that it is unclear to the callers what the system expects them to answer. Leading the customer through a dialog therefore also means asking concise and clear questions. LINGUINI is also able, in many cases, to **handle additional input** given by the user by triggering the Language Analysis component and searching for new information in GermaNet. This takes into account the fact that several information items may occur in a single user input.

In Section 3.7.1, we described different *knowledge sources* that play a role in dialog management. In our approach, we use a **dialog history** for preventing the same question from being asked several times. The **domain model** in our approach is the sociological Target Group model, which represents characteristic interests of user groups. We do not use a predetermined **task record**, since our task is to collect as much individual information as possible and to adapt to the individual user while doing so. GermaNet, although being a linguistically designed resource, also serves the function of a **world knowledge model** in our system. It supports the system in detecting semantic relatedness between interests. Since we allow system initiative only, we do not need to represent knowledge about **turn-taking or discourse obligations**. Finally, the **user model** is not only the desired result of the dialog but also plays an important role in dialog control.

3.8 Natural Language Generation

3.8.1 Natural Language Generation Approaches

Natural language generation (NLG) plays a role in those NLP systems which not only process but also produce natural language utterances themselves. In general, NLG investigates how high-quality natural language text can be produced automatically on the basis of a system-internal information representation [Hovy 1997]. Central topics investigated in the field of NLG are the human language production process (influenced by psycholinguistics), the planning and decision processes involved in NLG, and the strings that can be produced given a specific set of grammar rules [Busemann 2003].

There are various approaches to NLG depending on the application under consideration (e.g. machine translation, dialog systems, generation of text documents). NLG can be realized as complex systems which integrate various types of functionalities. According to Busemann [2003], such a complex NLG system has the following tasks:

- **Content selection:** The items which are to be verbalized are selected.
- **Document planning:** Semantic and rhetorical relations are made explicit between the items. The text structure is created.
- **Sentence aggregation:** Content items are assigned to individual sentences, such that repetitions are omitted and readability is warranted.
- **Lexical choice:** Specific words are chosen to represent content and meaning.

- **Referring expression generation:** The occurrence of new expressions is marked in the discourse, and references to previous occurrences are introduced.
- **Surface generation:** Morpho-syntactic methods for word order and inflection generate the final text through the use of grammatical knowledge.

These tasks are assigned to different system components, and it has to be taken into account that the tasks are not independent of each other. Three architectural types can be distinguished for existing NLG systems: sequential, interacting, and integrated architectures [Busemann 2003]. In a *sequential architecture*, the data flow is unidirectional, from content selection to surface generation. Many systems follow this approach, e.g. MUMBLE [McDonald 1983], TEXT [McKeown 1985], or STOP [Reiter et al. 1997]. An *interacting architecture* allows a bidirectional communication at predetermined points in the architecture. A system following this approach is PAULINE [Hovy 1988]. PAULINE defines “decision points” (e.g. topic selection, content of a sentence, organization of sentence parts) at which an interaction may take place. In case the form determining component reaches a decision point, it passes control back to the content planning component. In *integrated architectures*, content and form are not distinguished. Both are part of a single, hierarchical planning process. An example is KAMP [Appelt 1985].

Busemann [2003] mentions that NLG can also be realized with fewer functions in smaller frameworks, which are often sufficient for practical application. This fact is mirrored by the distinction between deep generation and flat generation. *Deep generation* can be characterized as knowledge-based, theoretically well-founded, and problem-oriented [Busemann 2003]. These approaches normally integrate application-independent linguistic, world and domain knowledge-bases as well as complex semantic representations. A detailed account of deep generation systems can, for instance, be found in [Bateman 1997]. *Flat generation*, on the other hand, can be characterized as “opportunistic”, corpus-based, and application-oriented [Busemann 2003]. These systems often use their own (rather small) grammar, tailored to the individual needs of a specific application, and prefabricated text parts (“canned text”) where possible. A basic system for the development of flat generation systems is TG/2 [Busemann 1996].

[Hovy 1997] distinguishes the following NLG system types, listed in order of increasing technical sophistication. *Canned text systems* are the simplest approach and are used in the majority of existing systems. Here, the system prints a string which does not change (e.g. an error message or warning). Although these systems are easy to create, they are also wasteful, because a different output string has to be formulated for each case. *Template systems* are useful if a message has to be produced several times with only slight modifications. This approach is particularly suitable for generating texts which are rather regular in structure (e.g. some business reports). *Phrase-based systems* use a top-down approach for expanding phrasal patterns. These patterns may either refer to a sentence or discourse. The first selected pattern has to match the top level of the input (e.g. [SUBJECT, VERB, OBJECT]). The next selected pattern has to match part of this structure (e.g. [DETERMINER, ADJECTIVES, NOUN]) and so on, until each pattern has been replaced by words. This approach is mostly used for generating single sentences. It is very robust, but can become difficult to manage with increasing size of the underlying grammar, which defines the expansion of phrasal patterns. Finally, *feature-based systems* are among the most sophisticated existing NLG systems. Each possible minimal alternative for an expression is represented by an individual feature. Features that characterize a sentence are, for instance, POSITIVE/NEGATIVE, QUESTION/IMPERATIVE/STATEMENT, PRESENT/PAST TENSE, etc. For generating an expression, the appropriate features are collected incrementally for each part of the expression until the entire expression is fully determined. The advantage of these systems is that the con-

cept is simple: any distinction in language is defined as a feature. A disadvantage is the maintenance of feature interrelationships.

Existing NLG systems and their development are, for instance, described in detail by Reiter and Dale [2000]. Further valuable information on available software tools, publications, and events can also be found on the homepage of the ACL (Association for Computational Linguistics) Special Interest Group on Generation (SIGGEN, www.siggen.org).

3.8.2 The Natural Language Generation Approach of LINGUINI

The NLG approach taken in LINGUINI is oriented towards a flat generation (cf. Section 3.8.1) in that the generation process is strongly tailored to the specific needs of our profiling system. Our approach is most closely related to the template-based systems mentioned above by employing a set of reusable question frames with variable slots.

In our approach, the NLG of a profiling question for the user consists of two steps:

1. **Content planning and generation of a Question Specification:** The Dialog Manager decides on the content of the next question and generates a Question Specification, which contains detailed information about the question to be generated.
2. **Surface generation:** On the basis of this Question Specification, the NLG component creates the grammatically correct, final question string. For generating correctly inflected word forms it may cooperate with a separate morphological component or tool.

After the final question string has been generated, it is transformed into speech output for the user, as is described in Section 3.2.2. When we address the functioning of the Dialog Manager (cf. Section 5.4), we will describe Step 1 (the content planning and generation of a Question Specification) in detail. In the current section, we focus on Step 2, i.e. the surface generation and its input, the Question Specification.

The Dialog Manager has created a Question Specification for the next question and passes it on to the NLG component. Besides some other information needed by the Dialog Manager for reacting to the next answer, this **Question Specification** contains

- The question frame (a natural language question which contains variable slots)
- The current settings of the variables (i.e. words), which are used to fill the slots in the frame

Question frame:

„**Steht *pObject* *pActivity* bei Ihnen mit dem Ort *tgLocation* in Verbindung?**“
(Is *pActivity* *pObject* connected with the location *tgLocation* in your case?)

Profile node:

[*pObject*=Urlaub, *pActivity*=machen, ...]
([*pObject*=holiday, *pActivity*=going on, ...])

Target group node:

[*tgObject*=Urlaub, *tgActivity*=machen, *tgLocation*=in den Bergen, ...]
([*tgObject*=holiday, *tgActivity*=going on, *tgLocation*=in the mountains, ...])

Figure 3.11: Simplified Question Specification with question frame and variable settings

Figure 3.11 shows an example of a Question Specification. The figure is simplified in that it only presents information which is relevant for the NLG component. The question frame at the top of Figure 3.11 contains three variable slots: pObject, pActivity, and tgLocation (the prefix “p” stands for “profile node” and “tg” for target group node). The Question Specification also contains the profile node and its most similar target group node, which together determine the values for the variable slots, as indicated by the variable prefixes “p” and “tg”. The final question generated from this Question Specification is shown in Example 3.8:

Example 3.8

“Steht Urlaub machen bei Ihnen mit dem Ort ‘in den Bergen’ in Verbindung?”
 (“Is going on holiday connected with the location ‘in the mountains’ in your case?”)

For the **surface generation**, the NLG component may cooperate with an existent morphological tool to inflect the words which serve to fill the slots, such that finally a grammatical natural language question is created. For this purpose the morphological tool can use the question frame as contextual information as well as the attribute names (e.g. activity, object, property). The values of these attributes serve as fillers for the variable slots in the question frame. The attribute names are related to specific syntactic categories and hence indicate what kind of grammatical inflection is required. An *activity* attribute, for instance, indicates that its value is a verb and has to be inflected differently than an *object* value, which is a noun, or a *property* value, which is an adjective.

Step 1, the content planning and generation of a Question Specification, has been fully implemented in the Dialog Manager component of our prototype. For Step 2, the surface generation, we have implemented a simplified approach in our Question Generation component. At the moment we do not use a morphological tool to generate a grammatically correct final question string. Our approach is designed, however, to allow adding such a component easily.

The presently implemented simplified approach consists in merely inserting the variable values into the question frame without inflecting them. The result is a question which is correct with respect to content, but in some cases is not yet a grammatically correct natural language expression. Example 3.9 shows some final question strings, which have been created by our prototype with this method. In case 1), the output sounds ungrammatical, but can be well understood. In case 2), the output sounds grammatically acceptable even without morphological processing. In fact, many of the questions generated by our prototype at the moment sound grammatically correct, which is mainly due to the formulations chosen for the frames.

Example 3.9

- 1) Sie haben erwähnt, dass Sie sich für Filme interessieren. Ansehen Sie Filme?
- 2) A propos Filme ansehen : Mögen Sie Science-Fiction?

Da Sie sich für Bücher interessieren, interessiert Sie auch das Thema Sachbuch?

With this simplified approach, we can demonstrate that questions with reasonable and correct content can be generated without the additional effort of implementing or integrating a morphological tool. For making the tool available for public use, however, the system should integrate a morphological component or tool as described above.

3.9 Summary

We may summarize that there are no existing systems which apply the above mentioned NLP methods and resources to user profiling in an application scenario like the one addressed in this thesis. Many authors (e.g. [Zukerman and Litman 2001, Bühler et al. 2002]) emphasize, however, that the combination of user profiling, personalization, and NLP is highly promising and that many of the NLP approaches described in this chapter could profit from a combination with user profiling. The approach of this thesis might therefore be especially interesting and relevant for the in-vehicle dialog applications discussed in Section 3.7.2.

We would like to conclude this chapter by summarizing again the main characteristics of our approach with respect to the NLP methods and resources applied. The central tasks of our profiling system, introduced at the beginning of the chapter (cf. Figure 3.1), can now be specified with respect to the above described technologies and concepts:

Answer analysis:

- **Speech Recognition:** The investigation of this topic was not in the focus of this thesis, but our system allows for the integration of a speech recognizer. We suggest investigating the use of a speech recognizer comparable to a dictation system like ViaVoice, which should integrate a feedback mechanism for unknown words and be trainable for background noise and the speaker's voice. Restrictions imposed by in-vehicle use have to be taken into account.
- **Tagging, Parsing, Lemmatizing:** For these tasks we use the stochastic TreeTagger together with its associated chunker. The main advantage is that we were able to obtain an already trained version which proved sufficient for our purposes.
- **Word Sense Disambiguation:** We have developed a hybrid disambiguation strategy based on 1) a modified Jiang and Conrath algorithm for calculating semantic similarity, 2) a modified Lesk algorithm for taking into account contextual information from other user profiles, and 3) a threshold-based strategy for clarification questions. To our knowledge, a combination of these three aspects is not yet used in existing systems.

Profile management and dialog management:

- **Lexical-Semantic Networks:** GermaNet constitutes the basis for our user model. It serves for handling not predefined user input as well as for dialog control (selecting semantically related questions). We use a subset of its relations and extend its graph structure by means of semantically complex interest nodes.
- **Dialog System:** This part constitutes a main contribution of our work. We developed our own dialog approach, which is characterized by the ability to find semantically related topics and by a flexible strategy to select questions adaptively to user interests. Further features are: system initiative, a mixture of free and predefined answer possibilities, the ability to handle additional user input, and the use of a dialog history and several dialog-related knowledge models (e.g. domain, world knowledge, and user model).

Question Generation

- **Natural Language Generation:** In our approach, NLG consists of two parts: 1) content planning (accomplished by the Dialog Manager) and 2) surface generation.

Surface generation is implemented at present by a simple approach, but this component may be extended for cooperating with a morphological tool.

- **Speech Synthesis:** The investigation of this topic was also not in the focus of this thesis, but our system allows for the integration of a speech synthesis. We suggest using an existing concatenative speech synthesis, which might use prerecorded versions of our question frames in combination with modeling continuous speech effects for smoothly integrating slot-filling words.

4 Overview of the Profile Acquisition Approach

The aim of this chapter is to give an overview of the LINGUINI profile acquisition approach, while Chapter 5 explains the involved components and methods in full detail. In Section 4.1, we summarize the main requirements for the profiling approach on the background of the investigations presented in Chapters 2 and 3. We then sketch the profiling process in Section 4.2 from a user's and a system-internal point of view. The LINGUINI system architecture is presented in Section 4.3. Section 4.4 finally introduces our Profile Model, which serves as a basis for creating user profiles and is fundamental for the other system components described in Chapter 5.

4.1 Requirements for the User Profiling System

The mobile application scenarios presented in Chapter 1 impose some central requirements on our approach. The most important requirement, which has been discussed in Chapter 1, is that the profiling process takes place while driving or walking around and should therefore be conducted by means of a natural language speech dialog.

Two main tasks have to be solved for achieving this goal: First, a suitable model for storing user information, methods for managing this information, and a concept for the profiling process have to be developed (Profile Manager). The second task is the development of a strategy, models, and methods for the natural language dialog with the user (Dialog Manager).

We summarize again the main requirements (1.-3.) for the **Profile Manager** (as introduced in Chapter 1), which we have augmented with some additional requirements:

1. The Profile Manager should store and maintain user information explicitly.
2. It should represent user interests and semantic relations between them in a structured way.
3. Long-term maintenance and exchange of user information among applications should be enabled. The exchange of information should be supported by a suitable modeling language (e.g. XML).
4. The interests stored should not be restricted to a specific application or domain (e.g. recommendation of tourist information only) or to a specific travel goal (e.g. the visit of larger cities). They have to characterize the individual user in a rather general way, such that other types of recommendations (e.g. books, music, gifts) can be generated on this basis as well.
5. The profile information has to be stored in a format which supports exchange between applications (e.g. XML).

The main requirements (1.-4.) for the **Dialog Manager** (as introduced in Chapter 1) as well as some consequential requirements (5.-7.) are listed in the following:

1. The dialog approach should be suitable for acquiring personal interests of users.

2. The Dialog Manager and its supporting components should be able to process and react to not predefined user input.
3. The dialog should adapt to the individual user's interests and related topics.
4. The solution should be usable with a speech interface in mobile situations or with typed input in other application scenarios.
5. The dialog interaction should be configurable for the needs of different applications. Configurable aspects should be, for instance, the overall duration of the dialog or specific topics or interest attributes which, because they are of particular relevance for the specific application, are favored for asking questions about in the dialog.
6. Some users may find it too demanding to be asked to enter free answers for each question. The dialog should therefore consist of both free answer questions and questions offering a predefined choice of values.
7. In cases where the dialog cannot adapt to the user input for some reason, the system should be able to suggest new topics that have a high probability of being interesting for the user.

In the remainder of this chapter and in Chapter 5, we explain how these requirements are fulfilled in our approach.

4.2 The Profile Acquisition Process

The **main aim** of the profile acquisition process is to learn about the user's individual interests and to store them in a semantically structured graph, in which relations between interests are represented. Each user interest corresponds to a node in the profile graph.

We concentrate on a subset of user information, **personal interests**, which are of particular relevance for recommender systems (cf. Section 2.1.2.4) and exhibit a complex semantic structure. This semantic structure is especially suitable for the acquisition by means of a natural language dialog, where different questions refer to different aspects of the structure. As a consequence, the profile node we create for an interest is a complex data structure, which consists of attribute-value pairs that further characterize this interest (cf. Section 4.4.2.1).

The user information is acquired explicitly during a **natural language dialog** interaction. A main characteristic of this dialog is that it **adapts** to the individual user, i.e. it asks refining questions about known interests of the user and investigates her interest in related topics. In the following, we illustrate the adaptive profiling dialog from a user's point of view, i.e. we mainly explain the features and characteristics of the process that are noticeable for the user. Some technical details (not directly noticeable for the user) will occasionally be discussed, however, if helpful for a better understanding.

4.2.1 The Process from a User's Point of View

The profiling process can be divided into two parts: the **initial profile construction** on the basis of some starting input entered by the user and the **refining of the profile** during the further dialog interaction with the user.

4.2.1.1 Constructing an Initial Profile

At the beginning of the profiling process, the user is asked a *starting question*, which allows for the initial construction of the profile. The formulation of the starting question as well as several other parameters are individually configurable by any application that wants to acquire user information (cf. Section 4.2.2.2). In the starting question, the user is asked to enter some interests in the form of phrases (headwords) (cf. Section 4.4.1), which for typed input are separated by semicolons. Example 4.1 shows a sample starting question and a possible user answer:

Example 4.1 (starting question)

Startfrage: “Geben Sie bitte 4 Ihrer persönlichen Interessen ein.”

(Starting question: “Please enter 4 of your personal interests.”)

Startantwort: “wandern mit meinen Kindern”; “Musik”; “öfters Filme ansehen”; “guter, alter Wein”

(Starting answer: “hiking with my kids”, “music”, “watching movies now and then”, “good, aged wine”)

4.2.1.2 Refining the Profile during a Dialog Interaction

After some initial information has been extracted from the starting answer and has been used for creating a profile, this profile can be extended to include further information by conducting a **natural language dialog** with the user. As explained in Chapter 3.7.3, only the system takes the initiative in our dialog approach. The allowed speech acts are also predetermined: the system only asks questions (except for the closing remark and occasional messages requesting changes in case of erroneous user input), whereas the user enters answers only. Users may nevertheless **influence the dialog flow indirectly** by mentioning new topics they find interesting in the answer to a more general system question (e.g. “Welche Themen interessieren Sie sonst noch (Which other topics are you interested in?)”). The system will then try to adapt by asking about these topics or related topics.

Broadening and Deepening Questions

The dialog is conducted according to a specific **questioning strategy**. The questioning strategy we took in the LINGUINI prototype is to explore related topics first, before asking refining questions about known interests of the user. Other possible strategies are discussed in Section 5.4.2.4. For realizing this strategy, the system uses **two main question types**, which either lead to new profile nodes or to further attributes added to existing nodes:

- **Broadening questions:** They ask the user about new topics which might be interesting for him. In case the user agrees, this leads to “broadening” the profile by creating new interest nodes.
- **Deepening questions:** They ask about details of already known user interests. This leads to “deepening” the profile by enriching existing interest nodes with further attribute values.

For both types, a variety of different questions can be generated. The topics that may occur in *broadening questions* are in most cases related to known interests of the user and are determined by means of the profile graph structure. For generating broadening questions, the hierarchic relations between profile nodes play a crucial role: hyperonyms, hyponyms, sibling

nodes, and nodes which occur in the same subtree of a defined height as a particular user interest are taken into account. Example 4.2 illustrates broadening questions. In Example 4.2 (1), the user has entered “Sport (sports)” and the broadening question generated in turn concerns its **hyponym** “Fußball (soccer)”. In Example 4.2 (2), a broadening question is asked about the **hyperonym** “Kunst (art)” of “Musik (music)”:

Example 4.2 (broadening question)

(1) The user has entered “Sport (sports)”. A possible broadening question is: “Da Sie sich für Sport interessieren, interessiert Sie auch das Thema Fußball (Since you are interested in sports, are you also interested in soccer)?”

(2) The user has entered “Musik (music)”. A possible broadening question is: “Da Sie Musik erwähnt haben, interessieren Sie sich auch für Kunst allgemein (Since you have mentioned music, are you also interested in art in general)?”

Deepening questions, on the other hand, try to elicit details about an already known interest. They always refer to attributes that can occur in the profile node structure, such as location, participants, activity, etc. These attributes are described in detail in Section 4.2.2.1. In Example 4.3, a deepening question is asked about the attribute “property = klassische (classical)” with respect to “Musik (music)”:

Example 4.3 (deepening question)

The user has entered “Musik machen (making music)”. A possible deepening question is: “Zum Thema Musik machen: Interessiert Sie speziell klassische Musik (With respect to making music: Are you interested especially in classical music)?”

Besides reacting adaptively to what the user has said, the system may also introduce completely new topics that are not yet related to some user interest. It selects these topics by comparing the user profile with an integrated model of user groups, from which it selects the group with interests that are most similar to the interests of the user (cf. Section 4.2.2.2). Since the group interests are similar to the known interests of the user, the probability is high that the user is also interested in the suggested new topic selected from this group.

Answer Types

For all questions generated by the system, we can distinguish **three possible answer types**:

- **Free user answer:** This answer type is syntactically restricted to phrases, i.e. no sentences or subordinate clauses are allowed. An example for a free user answer is “wandern mit meinen Kindern (hiking with my kids)”.
- **Selected value answer:** This answer type consists of a value the user selects from a value set suggested by the system, e.g. “sehr wichtig; wichtig; weniger wichtig (very important; important; less important)”.
- **“Ja/nein (yes/no)” answer:** This answer merely consists of “ja (yes)” or “nein (no)”.

A Sample Dialog with LINGUINI

Figure 4.1 shows a cutout of a profiling dialog with LINGUINI on the basis of the starting answer “Karate (karate); Theater (theater)”:

Startfrage: Geben Sie bitte 2-3 Ihrer persönlichen Interessen ein (getrennt durch “;”).

Startantwort: Karate; Theater

Frage (1): Wie häufig beschäftigen Sie sich Sie mit dem Interesse Theater ? (regelmäßig, häufig, gelegentlich, selten)

Antwort: selten

Frage (2): Mit welcher anderen Person gehen Sie dem Interesse Theater nach? (Ansonsten geben Sie bitte 'alleine' ein.)

Antwort: mit meiner Freundin

Frage (3): Welche/s/n Lieblings- Theater haben Sie?

Antwort: Burgtheater

Frage (4): Welche Dinge, Themen oder Aktivitäten interessieren Sie noch?

Antwort: kochen; Musik; Tiere

Frage (5): Da Sie sich für Tiere interessieren, interessiert Sie auch das Thema Pferde ? (ja, nein)

Antwort: nein

Frage (6): Da Sie Karate erwähnt haben, interessieren Sie sich auch für Kampfsport allgemein? (ja, nein)

Antwort: ja

Frage (7): Da Sie Kampfsport erwähnt haben, interessieren Sie sich auch für Sport allgemein? (ja, nein)

Antwort: nein

Frage (8): Da Sie sich für Theater interessieren, interessiert Sie auch das Thema Volkstheater ? (ja, nein)

Antwort: nein

Frage (9): machen Sie Musik ? (ja, nein)

Antwort: ja

Frage (10): Zum Thema Musik machen : Interessiert Sie speziell klassische Musik ? (ja, nein)

Antwort: ja

Frage (11): Welche/s/n Lieblings- Musik haben Sie?

Antwort: Rock

Frage (12): Mit welchem Ort steht für Sie Tiere in Verbindung?

Antwort: Wald

Frage (13): Wie häufig beschäftigen Sie sich Sie mit dem Interesse Tiere ? (regelmäßig, häufig, gelegentlich, selten)

Antwort: selten

Frage (14): Welche/s/n Lieblings- Tiere haben Sie?

Antwort: Hasen

[...]

Frage (19): Interessieren Sie sich für Radfahren?

[...]

Figure 4.1: Cutout of a sample dialog with LINGUINI

Normally, the next system utterance after the starting question would be a broadening question according to our strategy introduced above. In the sample dialog shown in Figure 4.1, however, “Theater (theater)” has been defined by the application designer as a topic with

particularly high priority for the application domain². As a consequence it is favored in the dialog. Questions 2 and 3 are hence deepening questions with respect to the interest “Theater (theater)”. Question 4 is a broadening question that allows the user to introduce new topics. Questions 5 – 8 are broadening questions about topics that are related to user interests. Here, Questions 5 and 8 ask about hyponyms (subordinate nodes) of the user interest, whereas Questions 6 and 7 ask about hyperonyms (superordinate nodes). Questions 9 – 11 are deepening questions about “Musik (music)” mentioned by the user in Answer 4. Questions 12 – 14 are deepening questions about “Tiere (animals)” mentioned also in Answer 4. In Question 19, the system introduces a completely new topic based on the interests of the most similar user group. Further sample dialogs taken from our user study can be found in Appendix B.3.

User Guidelines

For users who are unfamiliar with the system, it is helpful to make available **user guidelines**. This may not only help to support the user in getting ideas about how to conduct a profiling dialog with LINGUINI, but also for communicating system restrictions which require the cooperation of the user. User guidelines for LINGUINI should at least contain instructions for handling the system (e.g. the command for prematurely ending the dialog), examples of allowed user input or a cutout of a sample dialog, and possible restrictions of the system (e.g. that each user interest has to be clearly separated from the next one by a defined separation command, e.g. “;” in the typed-input variant). A speech version of these guidelines can be recorded in advance and read to the user while driving. The current user guidelines of the LINGUINI prototype system are discussed in connection with the system evaluation in Section 6.3 and can also be found in Appendix B.2.

4.2.2 The Process from a System-Internal Point of View

We will now give an overview of the construction and refinement of a user profile from a system-internal point of view. We describe the processing steps that underlie the profiling process and the responsible components and their interaction. A detailed account of the central system components will be given in Chapter 5.

4.2.2.1 Constructing an Initial Profile

For a new customer, the profile instance is initially empty. After the user has answered the starting question (cf. Example 4.1 above), each element of the answer (i.e. each separate interest) is linguistically analyzed and specific Attribute Extraction Rules are applied. A profile node is built from the attribute-value pairs found in the answer. These tasks are accomplished by the **Language Analysis** component (cf. Section 5.1). Afterwards, the **Profile Manager** (cf. Section 5.2) searches for this profile node in GermaNet and constructs a profile graph from the hyperonym relations found in GermaNet. In case a profile node cannot be found in GermaNet, it is stored temporarily under a special node in the profile called “NOT FOUND node”. The interest which has not been found can be relocated later during the dialog.

² The configuration of nodes with high application priority is addressed in Section 4.2.2.2 (“configurable parameters”).

4.2.2.2 Refining the Profile during a Dialog Interaction

After the initial profile has been constructed, it can be refined by conducting a natural language dialog with the user. The component which controls the dialog interaction with the user is the **Dialog Manager** (cf. Section 5.4). Its main task is the selection of a suitable next question adaptively and individually for each user.

During the question selection task, the Dialog Manager is supported by the **Target Group Matching** component, which uses a model of user groups, the so-called **Target Group Model** (cf. Section 5.3). This model contains **target groups** that represent specific user groups according to their personal interests. The responsibility of the Target Group Matching is to compare the user profile with the Target Group Model and to assign the user to the target group with the most similar interests. During the dialog, the Target Group Model serves as a repository for related topics that are interesting for particular user groups. The target groups have a similar structure to the user profiles such that they can be compared easily. If a specific relation (hyperonym, hyponyms, siblings, etc.) can be detected between a profile node and a target group node, a **broadening question** is asked about this target group node. If a profile node is identical with a target group node, **deepening questions** are asked about it. After each user answer, the modified profile is compared again with the Target Group Model and the next suitable question is selected.

In Section 4.2.1.2, we mentioned that besides asking broadening questions about related topics, the system also has the ability to **introduce completely new topics** itself. In case no node can be found in the Target Group Model that is closely related to one of the profile nodes, an unrelated interest can be selected from the most similar target group. If the actual user profile is most similar to the target group “Unterhaltungsmilieu (entertainment target group)” (cf. Sections 2.1.1.4 and 5.3.1.2), for example, the user can be asked whether she likes watching TV, because watching TV is highly characteristic for this group. This way, the system can also offer new topics to the user instead of only requesting them from the user.

Processing Steps during a Dialog Unit

A dialog consists of several dialog units. A *dialog unit* is the phase between a received user answer and the next system question. Figure 4.2 shows the system-internal processing steps during a single dialog unit. The process always takes into account the actual status of the information stored in the user profile.

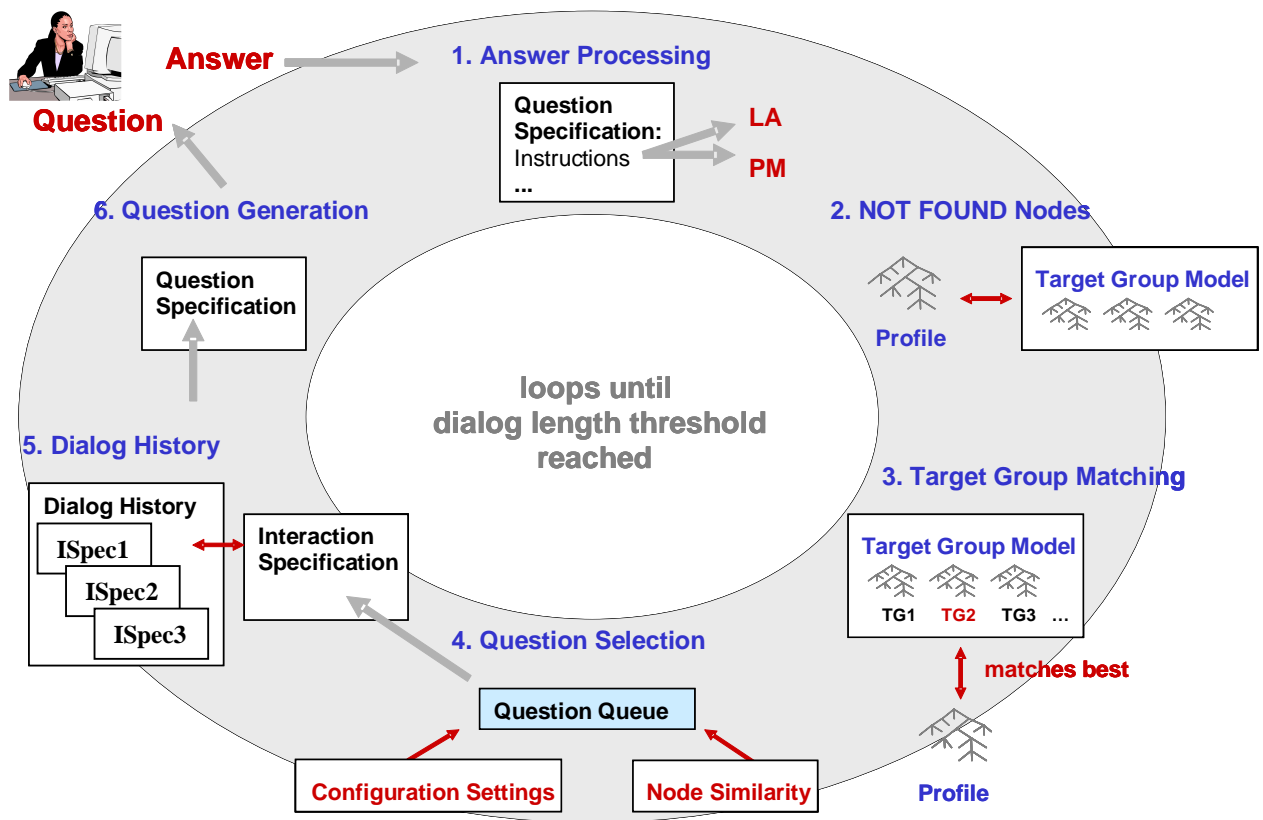


Figure 4.2: System-internal steps during a single dialog unit

In the following, we give an overview of the tasks accomplished during these processing steps shown in Figure 4.2. The full details are explained when discussing the system components in Chapter 5. The system components that accomplish each processing step are indicated in brackets:

1. [Dialog Manager, Language Analysis, Profile Manager] Answer processing: After the system has received a user answer, the Dialog Manager decides how to proceed on the basis of the *instructions* contained in the Question Specification (cf. Section 3.8.2) of the last question. Instructions concern the components Language Analysis and Profile Manager and determine how to handle the different types of answers (free answer, selected value, yes/no). If the user has entered a free answer, for example, the instruction is to analyze it linguistically (Language Analysis) and to insert the extracted information into the profile (Profile Manager). If the user has entered a yes/no answer, for example, the instruction is to insert the information into the profile directly (Profile Manager) without the necessity of a linguistic analysis.

2. [Dialog Manager] Relocation of NOT FOUND nodes: The aim of this step is to find the semantically right position in the profile graph for user interests that could not as yet be found in GermaNet and have been stored temporarily under the NOT FOUND node. This is done by exploiting related information stored in the Target Group Model. Sometimes, the object attribute of a user interest that has not been found in GermaNet (e.g. “Bildzeitung”) can be found as *favorite_items* attribute in the target groups. The object attribute of the target group node (e.g. “Zeitung”) can then be used to semantically locate the profile node in the graph. This process is described in detail in Section 5.4.2.2.

3. [Target Group Matching] Target Group Matching: The next step is the Target Group Matching. Here, the actual state of the user profile is compared with each target group

in the Target Group Model and the most similar target group is determined. This is done by comparing each node in the profile with each node in each target group. The similarity between profile and target group is calculated by taking into account the semantic relations between node pairs (e.g. hyperonym, hyponym, siblings).

4. [Dialog Manager] Question selection and creation of a question queue: After a most similar target group has been determined, the next step is the question selection. The question selection is the main process within the Dialog Manager and operates on the result of the Target Group Matching, i.e. on all profile-target group-node pairs contained in the most similar target group(s). Its task is to select from these node pairs the node which is most suitable for asking the next question. The most suitable target group node is determined by taking into account the semantic relation between a target group node and a profile node (contained in a node pair). A node pair with hyperonym relation (which leads to a broadening question) is, for example, preferred to a node pair with identity relation (which leads to a deepening question) (cf. Section 5.4). Moreover, the attribute values of both nodes are compared. In case both target group node and profile node contain the object “music”, but the target group node contains an activity “making” in addition, this additional attribute can be asked about in the next question (cf. Example 4.4). The **configuration settings** also play a role in this question selection process (cf. end of this section). Example 4.4 illustrates the influence of Target Group Matching and Target Group Model on the question selection process. Note that this is a simplified view on the more complex question selection process, which is described in detail in Section 5.4:

Example 4.4

Assume that the only known interest in the user profile is “music” and that Target Group 1 is characterized by an interest in “making music”, whereas Target Group 2 prefers “listening to music”. Both target groups have the same semantic similarity to the user profile at this stage of the profiling process, by means of the accordance “music”. The system would therefore take the first of the two target groups and select the most suitable node, “making music”, and would ask the user whether she makes music herself. After the user has answered, the Target Group Matching would calculate newly the most similar target group. In case the user answered “yes”, a similarity shift to Target Group 1 would be detected. In case the answer was “no”, a higher similarity to Target Group 2 would be detected now. The next question is then selected on the basis of this newly calculated most similar target group.

After the question selection has determined the node pair which provides the contents for the next question, it typically creates one (and in some cases two) **Question Specifications** for this node pair. A Question Specification contains all kinds of question-related information, e.g. the question frame, variable settings for filling the frame, instructions on how to react to the answer, etc.³ All Question Specifications created for the next question are stored in an **Interaction Specification**.

The Dialog Manager does not only plan one possible question that can be asked in the next dialog unit, but several (presently ten) questions, which it keeps for the case that the mostly preferred Interaction Specification has already been used during the dialog (cf. Step 5 “Dialog History Check” below). For this purpose, the Dialog Manager selects further node pairs about which questions could be asked. For each such node pair, an Interaction Specifica-

³ The term Question Specification has been introduced in Section 3.8.2 and will be explained in full detail in Section 5.4.2.1.

tions is created and stored in a **question queue**. In the queue, the Interaction Specifications are ordered according to their suitability for asking the next question

5. [Dialog Manager, Dialog History] Dialog History Check: A Dialog History keeps track of all Interaction Specifications that have been created and used during the dialog. This ensures that no identical question is asked twice. The next step is therefore to take the first, i.e. the mostly preferred, Interaction Specification from the question queue and search for it in the Dialog History. If it is contained there already, the next Interaction Specification is taken from the question queue, and so on, until an Interaction Specification is found that has not yet been used. In case the Interaction Specification under consideration does not yet occur in the Dialog History, the first Question Specification contained in it is passed to the Question Generation.

A special case occurs, if the Interaction Specification contains two Question Specifications, i.e. possible questions, and the first one has already been asked to the user. An Interaction Specification only contains two Question Specifications if the second question semantically depends on the answer of the first one, i.e. the second question might not be asked in isolation. In this case, the Dialog Manager checks whether certain preconditions are met in the user answer (e.g. whether the first question has been negated by the user) such that the second Question Specification can be used for creating another question about the same topic (cf. Section 5.4.2.2).

6. [Question Generation] Question Generation: The Question Generation now constructs the final natural language question on the basis of the Question Specification it has received. For this purpose, it instantiates the variable slots in the question frame with attribute values of the selected node pair. After this, the final question is asked to the user.

The dialog unit loop, which consists of these six steps, continues until the contingent of fallback questions have been asked (cf. Section 5.4.2.4) or at most until the configurable threshold for the maximum number of questions is reached.

Configurable Parameters of the Profiling Dialog

In Figure 4.2 above, we mentioned “configuration settings” in connection with Step 4, which may have influence on the question selection process and on the entire dialog. Their function will be described in this section.

One of our aims was that the LINGUINI profiling approach should be applicable to different applications for recommending different types of information, events, or products. In order to keep the dialog adaptable to the needs of different applications, we integrated configurable parameters into the system on two levels:

- **“Black box” parameters:** These may be configured easily by the application designer without an in-depth knowledge of the internal functioning of the system components.
- **“Glass box” parameters:** These parameters are for fine-tuning the functioning of specific system components. They should only be changed by experienced developers who are familiar with the technical details of and interdependencies between the components.

In the following we describe the **black box parameters** in more detail. The glass box parameters are explained in connection with the detailed description of the individual components in Chapter 5. Altogether, there are six configurable black box parameters. The following four black box parameters affect the Target Group Model, the duration of the dialog, as well as the formulation of greetings and other standard messages printed by the system:

- **Target Group Model IDs:** Since an arbitrary Target Group Model, which meets the needs of a specific application, can be used in LINGUINI, the IDs of the target groups that are supposed to be contained in it can be specified in the configuration file. Like user profiles, the target groups are stored in a database and each has a unique ID. In case the application designer uses a Target Group Design Tool (cf. Section 5.3.1.1), the IDs could be set automatically in the configuration file.
- **Dialog length:** This parameter determines the overall duration of the dialog. Its value may be set to an arbitrary integer > 0 . For our prototype evaluation (cf. Section 6.3), the value has been set to 30, i.e. the system asks at most 30 questions. Note that the dialog may terminate prior to this value, in case the user deliberately ends the dialog or the contingent of fallback questions has been used up and no further meaningful question can be asked (cf. Section 5.4).

The dialog length is best determined individually by the application designer according to the situation in which the information is supposed to be acquired. On a long highway trip in a vehicle, for example, more questions might be accepted by the user as opposed to a short online shopping session.

- **Starting question and closing remark:** The value of both parameters is a string of arbitrary length. The starting question begins the interaction with the user and may contain, for instance, a greeting and a first question (cf. Example 4.1 above). The closing remark concludes the dialog and may, for example, consist of a greeting, thanking the user, and the information on how or where to find the created profile (cf. Example 4.5). The formulations may be adapted to different application domains and types of customers.

Example 4.5 (closing remark for typed input and output)

“Vielen Dank für Ihre Zeit. Der Dialog endet jetzt. Die Informationen wurden Ihrem Benutzerprofil hinzugefügt, das Sie unter <Link> ansehen können.”

(Thank you very much for your time. The dialog ends now. The information has been added to your user profile, which you can find here: <link>.)

- **System messages:** LINGUINI furthermore offers the possibility to store system messages that provide the user with important information. Such information may refer, for example, to system usage, errors, together with advice on how to proceed, or explanations in case a certain system action takes longer than others (e.g. loading the knowledge base at the beginning of the dialog).

With the following two black box parameters, the application designer can influence the priority of interests and hence the order of selected questions:

- **Nodes with high application priority:** The application designer may specify interest nodes she selects from GermaNet that should be asked in the dialog with a higher priority than others. For a vendor of sporting goods, for example, all hyponym nodes of “sports” are probably more relevant than nodes related to other topic areas. These nodes can be stored in the configuration file with the most important node at the top of the list and the least important one at the bottom (cf. Example 4.6). If possible, the system asks about these nodes prior to others, taking into account their priority order (cf. Section 5.4.2.4).

Example 4.6

```
<nodePriorities>
  <Skifahren id="nGeschehen.2430.Skifahren"/>
  <Fußballspiel id="nGeschehen.2433.Fußballspiel"/>
  <Badminton id="nGeschehen.2428.Badminton"/>
  ...
</nodePriorities>
```

- **Attributes with high application priority:** As was the case with nodes, specific attributes may be specified that are of higher relevance for an application than others. For a recommender of travel options, for example, the attributes `activity`, `location`, and `participants` may be of higher relevance than others (cf. Example 4.7). A detailed description of these attributes can be found in Section 4.4.2.1.

Example 4.7

```
<attrPriorities>
  <attr>activity</attr>
  <attr>location</attr>
  <attr>participants</attr>
  ...
</attrPriorities>
```

4.3 The System Architecture

This section introduces the architecture of our profile acquisition system. The components that are involved in the profiling process are shown in Figure 4.3 below. The central components on the processing level (marked in grey) are the Dialog Manager, the Target Group Matching, the Profile Manager, and the Language Analysis.

The **Dialog Manager** communicates with the Dialog Interface on the presentation level to output questions to the user and to receive user answers. It communicates with the Language Analysis in order to have answers analyzed linguistically, and sends the acquired user information to the Profile Manager. The Dialog Manager communicates with the Target Group Matching, which provides helpful information for selecting the next question. A queue with suitable questions for the next dialog unit is created by the Dialog Manager and the most suitable question in this queue (i.e. its Interaction Specification) is passed to the Question Generation. The Question Generation is responsible for the final generation of natural language questions for the user. The Dialog Manager cooperates with a Dialog History and takes into account the configurable dialog parameters.

The **Target Group Matching** has the task to support the dialog control by matching user profile information with target group information. It calculates the target group that is most similar to the current state of the user profile. For this purpose it uses a Target Group Model, in which user groups are represented by means of characteristic interests.

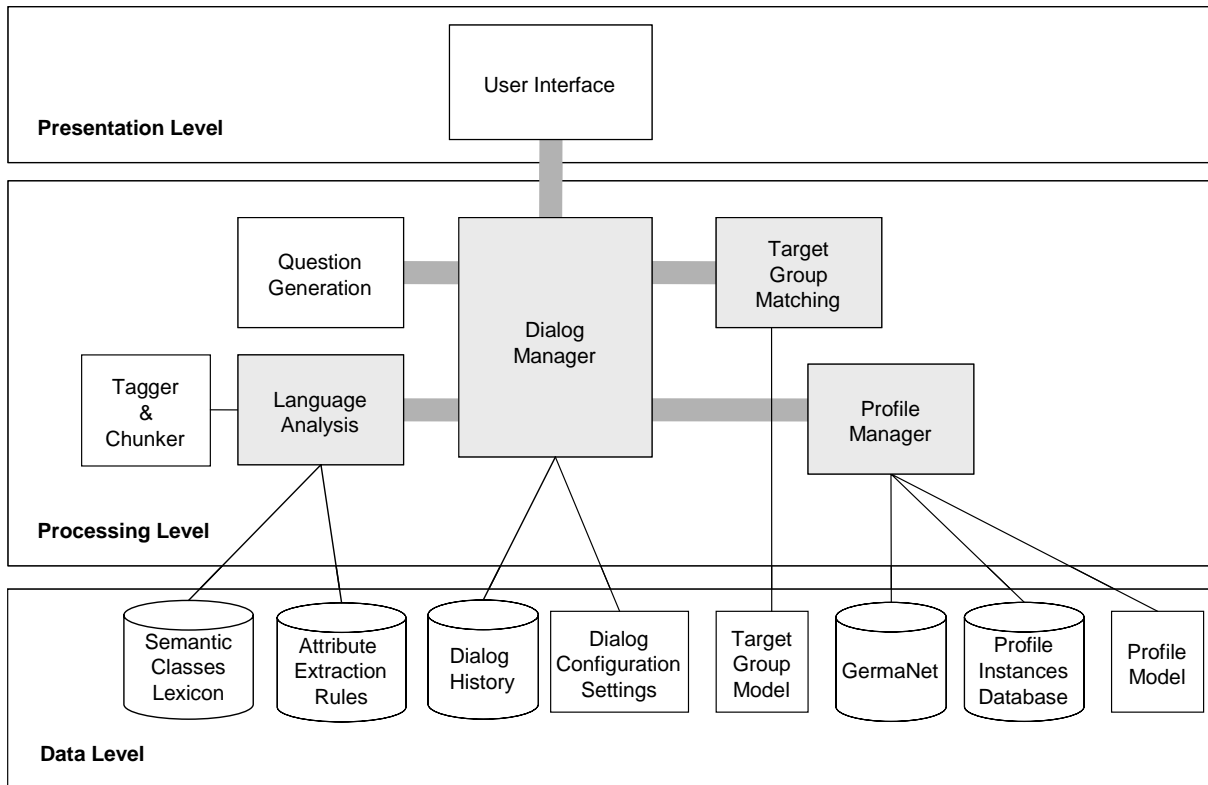


Figure 4.3: The LINGUINI system architecture

The **Profile Manager** is responsible for constructing an initial profile for a new user and for augmenting existing profiles with new nodes, attribute values, or attribute-value pairs. It uses the NLP resource GermaNet in order to find out where to semantically locate new information in the profile. The Profile Model specifies the structures and data types a profile may consist of. The Profile Instances Database stores the user profiles on a central server.

The **Language Analysis** uses external NLP tools (Tagger and Chunker) and a set of rules, which are used for the extraction of attribute values from user answers. Moreover, it uses a Semantic Classes Lexicon for checking semantic properties of words. The Language Analysis is called by the Dialog Manager for a specific user answer, which it analyzes linguistically. From the analyzed answer, the Language Analysis extracts attribute values and constructs one or several profile nodes, one for each user interest, and sends them back to the Dialog Manager.

These four central components will be described in detail in Chapter 5. But first, in the following section, we introduce our Profile Model which is fundamental for the functioning of the central system components.

4.4 The Profile Model

In our approach, we distinguish the Profile Model and profile instances (cf. Definitions 2.3 and 2.4 in Section 2.1.1.1). A *profile instance* refers to a specific customer and contains her personal data, whereas the *Profile Model* defines the data structures that are allowed in profile instances in general. Because an individual profile instance is created for each user, these instances may vary considerably, not only with respect to the type of interests stored, but also with respect to their structure.

In Section 2.1.2, we have justified the most important characteristics of our Profile Model, such as the focus on personal interests as a subset of user information and the explicit approach to user modeling chosen here. In this section, the Profile Model is presented in more detail. It has been developed on the basis of an empirical user study, in which users were asked to formulate their personal interests in natural language without giving any formal or syntactic restrictions to them. We have manually analyzed these user interests with respect to their syntactic and semantic properties in order to find an underlying information structure and structural synergies between different user interests. The user study is described in detail in Section 6.1. Besides the empirical data, the second major influence on the Profile Model was determined by the Dialog Manager's need for a linked or networked representation of user interests. The Dialog Manager needs such a representation for determining semantic relations between interests and finding related topics during the dialog. While the empirical study mainly influenced the attribute-value pair structure of the complex interest nodes, the need for a linked representation determined the choice of a graph structure based on a lexical-semantic network such as GermaNet.

In the following, we first explain which types of interests (and dislikes) may be contained in profile instances (Section 4.4.1). In Section 4.4.2, the data structure of interest nodes and the profile graph structure are discussed.

4.4.1 Interests and Dislikes Contained in Profile Instances

We will first present some examples of interests that have been formulated by users in the empirical study (cf. Section 6.1). These types of interests also correspond to the type of user answer we expect during the profiling dialog (cf. Example 4.8):

Example 4.8

Logik	logic
Tango Argentino	Tango Argentino
felsklettern	rock climbing
visuell schöne Gegenstände	visually beautiful objects
Entwicklung von Galaxien	development of galaxies
Rad fahren	cycling
ganz viel Sport	a lot of sports
schwimmen	swimming
Freunde treffen	meeting friends
den Osten erkunden	exploring the East
spielen mit meiner Tochter	playing with my daughter

In the empirical study, users only formulated their interests, not their dislikes. Dislikes, however, may also be taken into account by recommender systems to avoid generating recommendations which the user would reject. Therefore, we extended our approach to also be able to acquire and handle **dislikes** of users such as the following (Example 4.9):

Example 4.9

keinesfalls tauchen in Australien	in no case diving in Australia
keine Krimis von Agatha Christie	no mystery stories by Agatha Christie
nicht schwimmen	not swimming
deutsche Schlager auf keinen Fall	in no case German pop songs
ins Theater gehen niemals	going to the theater never
Sprachen lernen nicht so sehr	learning languages not so much

We manually analyzed large sets of interests collected in the user study, which led to the generalization that an interest can be represented by means of a **complex semantic structure** that contains various characterizing details about the interest. These characterizing details, such as locations, participants, ratings, properties of objects, favorite items, etc. can be modeled by means of attribute-value pairs. Dislikes can be modeled in the same way, while marking them by means of the `dislike` attribute in addition. In the next section, we explain this complex node structure and also show how interests are linked with each other in the graph structure.

4.4.2 The Profile Graph and the Structure of Interest Nodes

The graph structure we assume in our Profile Model semantically groups user interests into topics and subtopics and mirrors the semantic relatedness between interests. The graph structure is partly based on the formalism of the lexical-semantic resource GermaNet. The technical terms with respect to GermaNet used in this section have been introduced in Section 3.5.1. As outlined in Section 3.5.2, our Profile Model graph is a **directed acyclic graph**. **Edges** of the graph are either **hyponym relations** (i.e. the relation x is *hyponym* of y is directed from child node x to parent node y) or **hyperonym relations** (i.e. the relation x is *hyperonym* of y is directed from parent x to child y) as defined in GermaNet. A **node** in the graph corresponds to a GermaNet synset. The **node name** is the first element (lexical unit) of the synset. A node is either a (semantically complex) user interest or a (simple) intermediate node in the path between interest nodes, which does not represent a user interest. A node is an *interest node*, if it contains at least one attribute-value pair. Interest nodes may occur at any position in the graph. *Intermediate nodes* (i.e. nodes without attribute-value pairs) can also be seen as topic areas or related topics that help to semantically locate interests and that can be asked about during the dialog. The status of being interest node or intermediate node is not static. It is possible for intermediate nodes to become interest nodes during the dialog, as the system finds out more about the user's interests. Then these nodes are enriched with attribute-value pairs.

An exception in this structure is the “NOT FOUND node”, which is used to temporarily subsume interest nodes that could not be found in GermaNet. The role of the NOT FOUND node is discussed in more detail in connection with Dialog Manager and Profile Manager and the process of locating profile nodes in the tree in Chapter 5.

Hence, a profile instance is a **subgraph of GermaNet**, which is extended with complex interest nodes and the NOT FOUND node. This implies that not all GermaNet synsets are contained in a user profile, only those that are relevant for modeling the interests of a particular user.

Figure 4.4 shows a sample cutout of a profile instance graph. Two sample interest nodes, “mit meinen Kindern in den Alpen wandern (hiking in the Alps with my kids)” and “spannende Bücher lesen (reading exciting books)”, are shown in the boxes. The other nodes in angle brackets (< .. >) are intermediate nodes building paths to the interests in the boxes. They merely consist of a node name (i.e. a GermaNet lexical unit), e.g. “Sport”, followed by a node ID, e.g. “nGeschehen.953.Sport”.

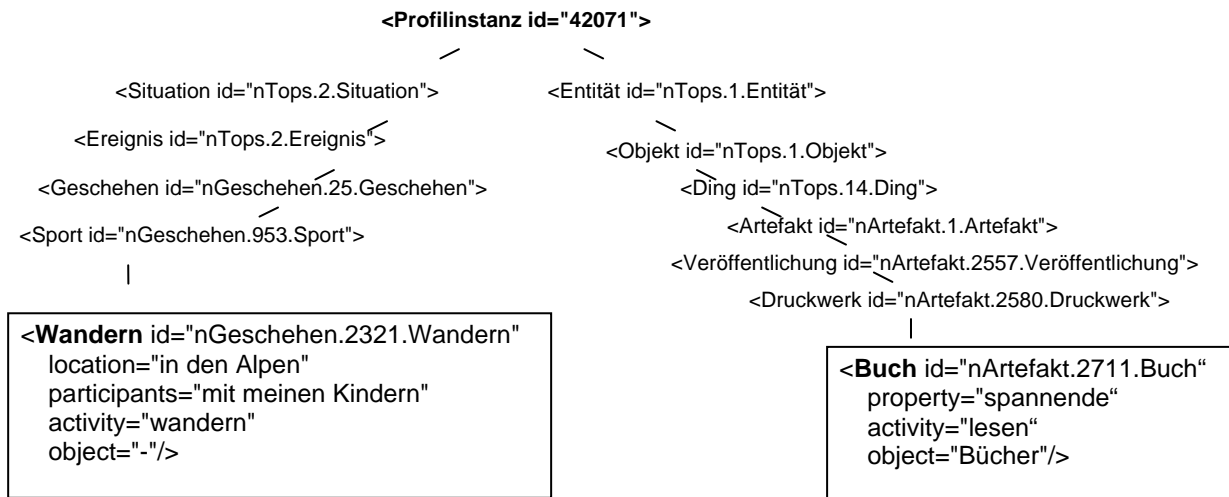


Figure 4.4: Cutout of a user profile instance⁴

4.4.2.1 The Internal Structure of an Interest Node

An interest node represents a specific user interest and consists of a **node ID** (for localizing the node in the profile graph), a **node name** (which is the first lexical unit of the corresponding GermaNet synset), and **attribute-value pairs**, which further characterize the interest. The node ID is the GermaNet-internal ID that belongs to each synset. The node IDs are extracted from GermaNet by the Profile Manager during the profile construction and are necessary for a unique reference to the node, since node names are not necessarily unique in the graph.

Example 4.10 shows two interest nodes. Node (1) represents the interest “lesen (reading)” in the case that no additional information is known. The node name, shown in the first line, is “Lesen (reading)”, followed by the node ID “nGeschehen.2415.Lesen”. Node (2) also represents “lesen” but with additional information given. Here, the node name is “Buch (book)” and the node ID is “nArtefakt.2711.Buch”. The name of a node is determined according to a specific algorithm described in Section 5.2.2.2.

Example 4.10

(1) User input: *lesen (reading)*

Interest node:

```
<Lesen id = “nGeschehen.2415.Lesen”
  object = “- ”
  activity = “lesen (reading)”/>
```

(2) User input: *häufig spannende Bücher lesen in meinem Garten, besonders Harry Potter*
(often reading exciting books in my garden, especially Harry Potter)

Interest node:

```
<Buch id = “nArtefakt.2711.Buch”
  object = “Bücher (books)”
  activity = “lesen (reading)”
  location = “in meinem Garten (in my garden)”
```

⁴ For reasons of readability, the original GermaNet terms have not been translated in Figures 4.4, 4.5, and 4.6.


```

frequency = "häufig (often)"
property = "spannende (exciting)"
favorite_items = "Harry Potter"/>

```

In Table 4.1 below, we present the **list of attributes** and their possible values⁵ that can occur in an interest node. This selection of attributes is based on the analysis of user data collected in the empirical study (cf. Section 6.1) and covers most input types that occurred in the study. It is theoretically possible to extend the Language Analysis component to be able to analyze other natural language input structures and extract further attribute types. However, the list of attributes presented here is sufficient for handling input structures comparable to those presented in Section 4.4.1.

Most of the attributes we use are determined by the **grammatical functions** (e.g. accusative object, main verb, etc.) their attribute values can have in the user input. The attribute `object`, for example, is typically the accusative object of a verb or the main noun in a noun phrase occurring without verb in the user input. The attribute `activity` typically is the main verb. Prepositional phrases are divided – according to their semantics – into locations, human beings, etc. Some attributes have values with a **clearly defined syntactic position** in the user input. The attribute `property`, for example, has as a value an adjective that occurs *before* the noun within a noun phrase. Another example is the attribute `nominal_modifier`, which in our approach is restricted to a prepositional phrase that occurs directly *behind* a noun and begins with a specific preposition (e.g. “aus” in “aus Plüsch (made from plush)”). The selection of attributes is not based on a specific linguistic theory, however, but rather empirically determined by our input data.

Attribute	Value	Status	Example
<code>object</code>	The value must be one of the following: 1. a noun phrase (in case the input does not contain a verb) 2. an accusative object (noun phrase) of a verb contained in the input 3. “-” (if no noun is contained in the input) In case 1. and 2., the value is always the inflected form occurring in the user input (e.g. “Autos (cars)”) and not the morphological base form (e.g. “Auto (car)”).	obligatory	Autos (cars); Bücher (books)
<code>nominal_modifier</code>	a phrase modifying the object	optional	aus Plüsch (plush) (e.g. in: “plush sofas”)
<code>activity</code>	verb representing an activity	optional	schwimmen (swimming)
<code>location</code>	place related to the interest	optional	Rotes Meer (Red Sea); im Garten (in the garden)
<code>participants</code>	participants of the activity; participants have to be human beings in our approach	optional	mit Freunden (with friends)
<code>property</code>	adjectival phrase modifying the object	optional	teure (expensive) (e.g. in: “teure Vasen (expensive vases)”)
<code>rating</code>	positive rating related to the interest; in our approach a rating typically expresses a positive degree of liking or interest	optional	besonders (especially)

⁵ Where a value of an attribute clearly corresponds to a phrase of a specific syntactic category, this is indicated.

frequency	frequency related to the interest; in our approach, a frequency typically is a prepositional phrase (e.g. “im Sommer (in summer)”) or adverbial phrase (e.g. “wöchentlich (weekly)”)	optional	im Sommer (in summer); wöchentlich (weekly)
dislike	negative rating related to the interest; in our approach a dislike typically expresses a degree of disliking or aversion	optional	nicht (not); keinesfalls (in no case)
favorite_items	proper names representing favorite items	optional	“Harry Potter” as favorite item related to the object “Bücher (books)”

Table 4.1 Attributes that may or must occur in interest nodes

The `object` attribute is obligatory, all other attributes are optional (cf. Table 4.1). In case an optional attribute does not have a value, its name does not occur in the interest node. The value type for all attributes is a set of strings. `Object` is the only attribute that is restricted to one string element at most. All other attributes can have zero or more values. In case the `object` is unknown, it gets assigned the value “-”.

4.4.2.2 Converting the GermaNet Graph into an XML Tree

In order to support the exchange of profile information between applications, we use XML as the formalism for representing user information. We therefore had to solve the conceptual problem of representing the original GermaNet graph as a tree, because a tree is the typical structure of an XML document. Figure 4.5 illustrates this problem by means of the interest “Fußball (soccer)”. It shows a small cutout of the graph representation of this synset in GermaNet:

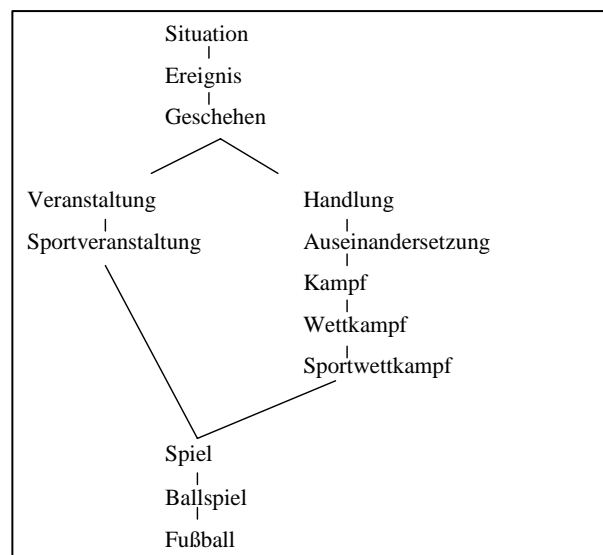


Figure 4.5: Cutout of the GermaNet graph for “Fußball (soccer)”

“Fußball (soccer)” can, for instance, either have the meaning of an event (“Veranstaltung”) or of an action the user performs (“Handlung”). Both event and action are in turn subsenses of the node “Situation (situation)”. The graph structure represents this relatedness by connecting the synsets (nodes) with each other via edges (cf. Figure 4.5).

We solved the problem of mapping the GermaNet graph into an XML tree by inserting “Fußball (soccer)” twice into the XML tree structure, but with identical node IDs (cf. Figure 4.6):

```

<Profilinstanz>
...
<Situation id="nGeschehen.1.Situation">
  <Ereignis id="nGeschehen.2.Ereignis">
    <Geschehen id="nGeschehen.25.Geschehen">
      <Veranstaltung id="nGeschehen.571.Veranstaltung">
        <Sportveranstaltung id="nGeschehen.938.Sportveranstaltung">
          <Spiel id="nGeschehen.1004.Spiel">
            <Ballspiel id="nGeschehen.2422.Ballspiel">
              <Badminton id="nGeschehen.2428.Badminton" object="Badminton"/>
              <Fußballspiel id="nGeschehen.2433.Fußballspiel" object="Fußball"/>
            </Ballspiel>
          </Spiel>
        </Sportveranstaltung>
      </Veranstaltung>
      <Handlung id="nGeschehen.1036.Handlung">
        <Auseinandersetzung id="nGeschehen.2004.Auseinandersetzung">
          <Kampf id="nGeschehen.2008.Kampf">
            <Wettkampf id="nGeschehen.952.Wettkampf">
              <Sportwettkampf id="nGeschehen.964.Sportwettkampf">
                <Spiel id="nGeschehen.1004.Spiel">
                  <Ballspiel id="nGeschehen.2422.Ballspiel">
                    <Badminton id="nGeschehen.2428.Badminton" object="Badminton"/>
                    <Fußballspiel id="nGeschehen.2433.Fußballspiel" object="Fußball"/>
                  </Ballspiel>
                </Spiel>
              </Sportwettkampf>
            </Wettkampf>
          </Kampf>
        </Auseinandersetzung>
      </Handlung>
    </Geschehen>
  </Ereignis>
</Situation>
...
</Profilinstanz>

```

Figure 4.6: Cutout of a user profile represented in XML

In Figure 4.6, the interest node “Fußball” occurs as a leaf node in the XML tree twice (in bold type), with the same node ID “nGeschehen.2433.Fußballspiel”. The intermediate nodes “Ballspiel” and “Spiel” occur twice in the graph too, with the same node IDs each. Further up in both paths for “Fußball”, we see that the paths differ in meaning: one contains nodes related to the sense of “Veranstaltung”, the other contains nodes related to the sense of “Handlung”. At the top of the tree, we have one common path again with the nodes “Geschehen”, “Ereignis”, and “Situation”. In summary, the strategy is to insert every node, which has n parents, n times into the tree. For $n > 1$, we express that the n nodes are identical in the original GermaNet graph by using the same node ID for them in the XML tree.

4.4.2.3 Using GermaNet Nouns for the Profile Graph

GermaNet is divided into nouns, verbs, and adjectives. All **node names** contained in our profile graph are a subset of the **GermaNet nouns**, i.e. the graph does not contain nodes from other syntactic categories stored in GermaNet. We took this design decision in order to keep the node names as homogeneous as possible. In GermaNet, the node names for verbs are completely different than the ones for nouns. Merging them would lead to a confusingly heterogeneous semantic structure, which could cause problems for interpreting and using the profile. Figure 4.7 gives an impression of the differences of the node names in the path of the example “wandern (hiking)”, which occurs both as noun and as verb in GermaNet:

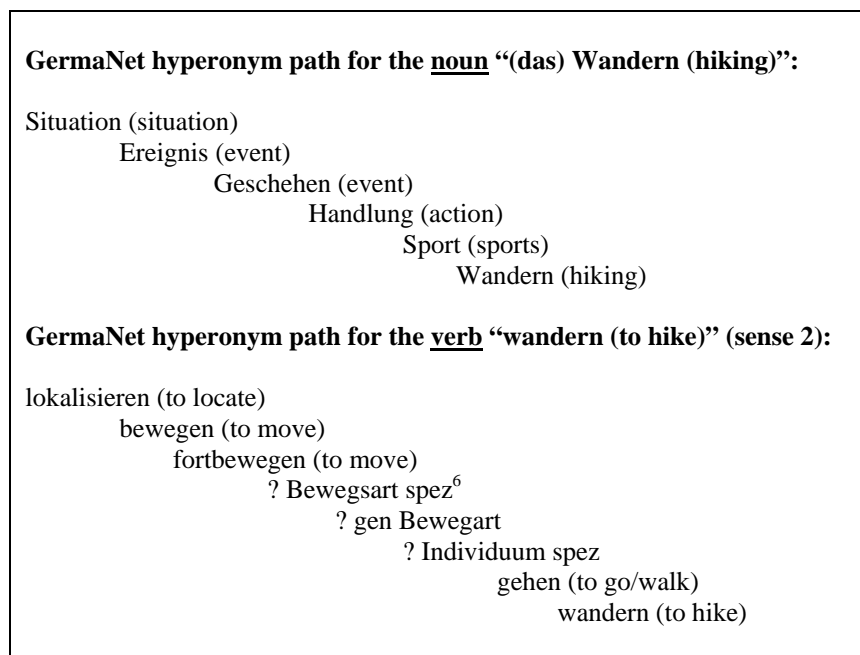


Figure 4.7: Different GermaNet paths for “wandern” realized as noun or verb

Finding interests that express activities in the GermaNet noun graph is not a problem theoretically, because the GermaNet nouns also include nominalized activities (e.g. “(das) Wandern (hiking)”). As a consequence, we can actually find most of the activities not only in the verb graph, but also in the noun graph. The “syntactic surface realization” of words as nouns or verbs only plays a role for the Language Analysis when extracting attribute values (cf. Section 5.1), but it is not crucial for localizing interest nodes in the graph. For localizing nodes, only the underlying semantics of the interest plays a role.

It is important to note that each user interest stored in the profile graph is represented not only by a single word form, but by a complete **synset**, i.e. a set of synonym word forms that all refer to the same meaning. The node name is the first element of this synset and serves only as an identifier. This implies that we can, for example, detect the identity between a profile node and a target group node which both represent the same meaning, even if their node names are different (cf. Example 4.11).

⁶ Node names with “?” refer to so-called *artificial nodes* used by the GermaNet developers for internal modeling purposes only. They do not correspond to real world concepts.

Example 4.11

Profile node name: “Lektüre (reading)” → Both are represented by the same synset
Target group node name: “Lesen (reading)” “{Lesen, Lektüre} (reading)” and can
be identified as **equal in meaning** al-
though their node names are different.

4.5 Summary

In this chapter, we presented an overview of our profile acquisition approach. We first summarized the main requirements for our profiling system, which were derived from the investigations presented in Chapters 2 and 3. We illustrated the profiling process and the type of dialog that is possible with our system. An important aspect is that the dialog is configurable in two respects. First, it can be configured by the application designer for adapting it to the needs of a specific application and, secondly, it can be configured by a system developer for fine-tuning the Target Group Matching and the question selection process. We discussed different types of questions with which the system adapts to the individual user input and introduces related and new topics during the dialog. We also explained how the user may influence the dialog flow. We presented the LINGUINI system architecture which has been developed from scratch for this thesis. With the architecture, we introduced the central system components and their interaction, with which we will continue in the next chapter. One of them is the Profile Model. The development of the Profile Model has been substantiated by an empirical user study, which we describe in Section 6.1. The Profile Model constitutes a main contribution of this thesis and its most important achievements are the following: 1) it models complex user interests that are relevant and desirable for many recommender applications and have not yet been provided by any other user modeling approach, 2) it supports the dialog control in determining semantic relations between interests and in finding related topics, and 3) its contents and knowledge representation are not bound to a specific application, but enable various types of applications to draw inferences on the basis of its semantic structure.

5 The Main Components of the Profile Acquisition System

This chapter is dedicated to the main system components that also constitute the main contribution of this thesis. We describe their individual composition and tasks as well as their interaction and embedding in the profiling system. First, the **Language Analysis (LA)**, which is responsible for analyzing natural language user answers, is described in Section 5.1. In Section 5.2, we explain how profile graphs are created and maintained by the **Profile Manager (PM)**. In Section 5.3, we describe the **Target Group Matching (TGMAT)** and the **Target Group Model (TGMOD)**. The TGMAT is fundamental for the adaptive question selection process. This process is finally described together with the most important system component, the **Dialog Manager (DM)**, in Section 5.4.

5.1 The Language Analysis

We first address the **Language Analysis (LA)** component. This section is structured as follows. Section 5.1.1 gives an overview of the LA. We describe the main tasks and the embedding of the LA. We also explain design decisions and additional problems we came across while designing the LA. The Semantic Classes Lexicon, a central resource we have created for the LA, is introduced.

In Section 5.1.2, the functioning of the LA component is discussed in detail. First, the LA classes are introduced. The remaining sections explain how the LA solves its main tasks: the linguistic analysis, the extraction of attribute values, and the creation of interest nodes.

5.1.1 Overview of the Language Analysis

5.1.1.1 Tasks and Embedding of the Language Analysis

The LA is called by the DM, when a natural language user answer has been received and needs to be analyzed linguistically. Figure 5.1 below shows the interface between these two components: the LA receives a user answer from the DM, i.e. a natural language string. This is the original string the user has entered. It may consist of one or several user interests. Example 5.1 shows a sample answer string, consisting of four interests:

Example 5.1

spannende Bücher lesen; Autos; tauchen im Roten Meer; mit Freunden kochen
(reading exciting books; cars; diving in the Red Sea; cooking with friends)

The task of the LA is to analyze this answer string, identify the interests contained in it, and transform each interest into the attribute-value pair structure of an interest node (or: profile node) (cf. Section 4.4.2.1). For the transformation of the analyzed answer into an interest node, the LA uses a set of Attribute Extraction Rules that try to find suitable attribute values

in the answer. From the attribute values found in the answer for a single user interest, the LA builds an interest node which represents this user interest. Finally, one or several interest nodes (each representing a single interest) are returned to the DM. This set of interest nodes constitutes the analysis result. Samples of interest nodes have been presented in Section 4.4.2.1.

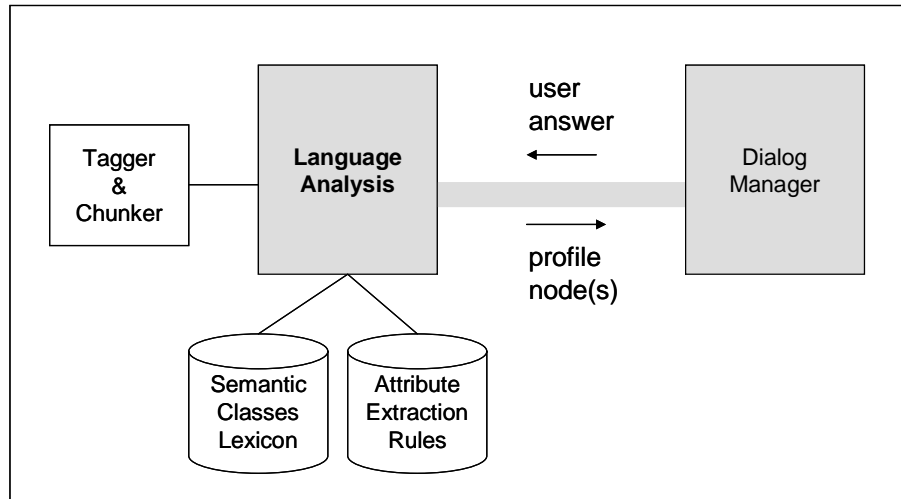


Figure 5.1: Interaction between Language Analysis and Dialog Manager

The LA accomplishes its task by employing the **TreeTagger**, an external part-of-speech tagger, and its associated chunker, which we have presented in Sections 3.3.2 and 3.4.2. The TreeTagger assigns part-of-speech tags to all words contained in the user answer and the chunker finds “chunks” in it, i.e. words that syntactically constitute a unit. Furthermore, the LA uses a set of **Attribute Extraction Rules** that are applied to the tagged and chunked user answer in order to extract attribute values for the interest node. The LA finally makes use of a **Semantic Classes Lexicon**. In this resource, the semantic class of a word (e.g. location, human being, time expression, rating) can be looked up. Hence, the tasks of the LA consist of the following processing steps, which we will describe in depth in the Sections 5.1.2.2 – 5.1.2.4:

1. Part-of-speech tagging and chunking of the user answer (by TreeTagger, chunker)
2. Applying the Attribute Extraction Rules for finding attribute values in the answer
3. Creating an interest node for each user interest from the attribute-value pairs

In Section 4.4, we explained that we conducted an empirical user study about preferred input structures (cf. Section 6.1) for the design of our Profile Model. The data collected in this study were also important for the design of the LA. For designing the LA, we linguistically analyzed the data manually in order to decide on the type of tagger and parser needed and on a method for extracting attribute values from the user input.

The statistical TreeTagger proved suitable for this type of input structure, because we only need to accomplish a shallow syntactic analysis (cf. Sections 3.3.2 and 3.4.2). The TreeTagger moreover has the advantage that it also provides base forms of words, i.e. it also functions as a lemmatizer. This capability is exploited by the PM for finding words in GermaNet and by the LA for looking up semantic properties of words in the Semantic Classes Lexicon. Since this lexicon has been derived from GermaNet (cf. Section 5.1.1.2), it also only contains base forms of words.

The analysis of possible user data also showed that the attributes needed for the construction of interest nodes (cf. Section 4.4.2.1) can be extracted by means of a small set of rules, which operate on the TreeTagger and chunker output, i.e. the syntactically analyzed user answer. These rules check whether certain preconditions in the syntactic structure and word order are given and on this basis extract specific words or phrases that serve as attribute values. A rule set of only nine rules (cf. Section 5.1.2.3) proved sufficient, because the input structures preferred by users were very homogeneous in the user study and of rather simple structure (cf. Sections 4.4.1 and 6.1).

The data collected in the user study also showed that syntactic analysis alone is not sufficient for extracting the attribute values we need. For some attributes, we need to check the semantic properties of words and phrases. Since we did not have access to a semantic lexicon resource, we created a simplified form of such a lexicon, which only contains the information we need, semi-automatically from GermaNet (cf. Section 5.1.1.2).

When designing the LA, we came across two linguistic problems that are not in the main focus of our work but should be mentioned here briefly. The first problem is that the occurrence of **negations** in the user input may sometimes lead to subtle ambiguities, which also depend on the position of the negation, as shown in Example 5.2:

Example 5.2

- | | |
|--|--|
| (1) <i>keinesfalls</i> tauchen in Australien | (<i>in no case</i> diving in Australia) |
| (2) tauchen <i>keinesfalls</i> in Australien | (diving <i>in no case</i> in Australia) |

A problem is, for example, to decide whether the negation (e.g. “keinesfalls” in Example 5.2) refers to the entire phrase or only to specific elements of the phrase. In general, this is not a trivial problem and would need a theoretically substantiated linguistic analysis and handling. The practical solution we chose is the assumption that the negation always refers to the entire interest and all information contained in it. In Example 5.2 this would correspond to the interpretation that the user does not like diving in general and in particular not in Australia. As shown in our final user study (cf. Section 6.3), users prefer to enter interests rather than dislikes in most cases. Therefore this problem, which only occurs for dislikes, does not seem to be a major problem in practice.

The second problem we detected is **ambiguities which occur in coordinations**, which is discussed in Section 5.1.2.4 together with the creation of interest nodes.

5.1.1.2 The Semantic Classes Lexicon

As mentioned above, we needed to find a way of checking semantic properties of words during the answer analysis process. Some attributes we want to extract require values that are of a particular semantic class. For the `location` attribute, for example, we have to make sure that it only gets assigned values that in fact belong to the semantic class of locations. Typically, such information is stored in a semantic lexicon. Since we did not have access to such a lexicon, we used the information about semantic classes stored in GermaNet in order to create a simple semantic classes lexicon semi-automatically. In this section, we describe how we created the lexicon and what problems we had to solve.

The lexicon is realized as an SQL database. In this database, we store each lexical unit together with a unique ID and a semantic class tag. Two examples for lexicon entries, each consisting of an ID, the lexical unit, and its semantic class tag, are shown in Example 5.3:

Example 5.3

ID 1327, München (Munich), LOC
ID 1558, wöchentlich (weekly), TEMP

The following **semantic class tags** are used in the lexicon (cf. Table 5.1). The names of the semantic class tags are derived from the names of the XML files in which the GermaNet data are stored (see below). They have been chosen for the Semantic Classes Lexicon according to the attributes that need a value of this semantic class. Further semantic classes are not required by our attributes and have, therefore, not been included. In the leftmost column of Table 5.1, the semantic class tags are listed, followed by a description of their meaning in the second column. The third column indicates the attribute which needs a value of this semantic class. The rightmost column gives examples of such attribute values.

Semantic Class	Description	Attribute	Example
LOC	denotes a location	location	Garten (garden)
HUM	denotes a human being	participants	Familie (family)
PART	class name for participants	participants	alleine (alone)
TEMP	denotes a temporal expression	frequency	öfters (often)
RATE	denotes a positive rating	rating	besonders (especially)
DIS	class name indicating a dislike	dislike	nicht (not)

Table 5.1: Semantic class tags for attribute values

The Semantic Classes Lexicon has been created in two steps. First, all lexical units of a specific semantic class (e.g. all locations) have been extracted from GermaNet automatically. This was supported by the fact that GermaNet already groups lexical units of specific semantic classes in separate XML files (e.g. human beings, locations, etc.). Figure 5.2 shows a cut-out of the GermaNet locations file:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE synsets SYSTEM "germanet_synsets.dtd">
<synsets>
...
<synset id="nOrt.151" lexGroup="Ort" wordClass="nomen">
<lexUnit Eigenname="nein" artificial="nein" id="nOrt.151.Residenz" orthVar="nein" sense="1" stilMarkierung="nein">
<orthForm>Residenz</orthForm>
</lexUnit>
</synset>
...
<synset id="nOrt.158" lexGroup="Ort" wordClass="nomen">
<lexUnit Eigenname="nein" artificial="nein" id="nOrt.158.Jugendherberge" orthVar="nein" sense="1" stilMarkierung="nein">
<orthForm>Jugendherberge</orthForm>
</lexUnit>
</synset>
```

Figure 5.2: Locations represented in a GermaNet XML file

From the GermaNet files, we extracted the lexical unit between the tags `<orthForm> ... </orthForm>` (cf. Figure 5.2) and stored them in a wordlist. Next, the lexical units were assigned a semantic class tag (LOC for location in this example), which we derived from the

name of the GermaNet file from which the lexical unit has been extracted (e.g. “nomen.Ort.64.xml”). We created separate word lists for each semantic class. The final wordlists contain lexical units and their semantic class tag. In the second step, we inserted the extracted wordlists into the database.

We were able to use GermaNet for the extraction of the classes LOC, HUM, and TEMP, but not for PART, RATE, and DIS. Ratings (e.g. “besonders (especially)”) (RATE) and keywords for dislikes (e.g. “nicht (not)”) (DIS) had to be inserted manually into the lexicon because they are mainly adverbs and particles, which are not contained in GermaNet. Furthermore we added manually to the lexicon a few keywords that indicate a “mode of participation” (e.g. “alleine (alone)”), and which we labeled with the semantic class tag PART. The words we had to add manually were very few (less than 30 overall) as compared to thousands of LOC, HUM, and TEMP expressions we could extract from GermaNet automatically.

The Semantic Classes Lexicon was created during the implementation phase of the profile acquisition system. Now that the lexicon has been created, it can be used for all further user interactions.

Problems with the Automatic Extraction of Wordlists from GermaNet

When constructing the Semantic Classes Lexicon, we came across several problems. The two main problems were missing words and misleading words. The problem of *missing words* is related to the fact that GermaNet is not yet a complete resource for the entire German language, but is still under construction. As a consequence, it does not contain all words of a specific semantic class that can occur as user input in our profiling approach. For example, GermaNet at present does not contain adverbs. Nevertheless, we did not want to restrict the user’s free choice of words. For this reason, we added manually to the lexicon the most typical words that occur within a semantic class in our user input and that are not contained in GermaNet. Here again, the empirical user study (cf. Section 6.1) provided us with information about typical words of a semantic class. For example, we had to insert certain typical adverbs manually, which serve as keywords for specific attributes (e.g. “alleine (alone)”, “besonders (especially)”). What we still cannot handle with this approach are *new word formations* users come up with. With new word formations, we mean expressions that are not yet contained in standard dictionaries for German. This problem is common for most NLP systems, however.

The second problem was that of *misleading words*. The lists extracted from GermaNet contain expressions which are not suitable representatives of the semantic classes in our sense. If they remain in the lexicon, this has the effect that some elements of the user input are wrongly considered attribute values, such as shown in Example 5.4:

Example 5.4

“Gedränge (abstract noun denoting the state or action of people standing closely together in a crowd and touching or pushing each other; may also refer to the crowd with these properties itself)” is a representative of the semantic class “human being” (HUM) extracted from GermaNet. This is because GermaNet also contains expressions denoting **groups of humans**. Since the word is of class HUM, it should be able to serve as value for the attribute participants. However, “Gedränge” is inappropriate as a participant of an activity, whereas friends, a sports club, etc. are more suitable.

A possible but linguistically difficult and costly solution would be to manually delete misleading words from the extracted lists. Reliable criteria would have to be found for distinguishing more suitable representatives from less suitable ones, because this is not always obvious even for linguists. We did not delete misleading words from the lists, since we wanted to show the feasibility of using such lists for attribute extraction in the first place.

5.1.2 Functioning of the Language Analysis

In Section 5.1.1.1, we introduced the three main tasks of the LA: the linguistic analysis of user input by means of the TreeTagger and its associated chunker, the extraction of attribute values, and the creation of an interest node for each user interest. In this section, we explain these tasks in detail. We start with an overview of the LA classes and their interaction.

5.1.2.1 Language Analysis Classes

Figure 5.3 shows the classes and resources of the LA that implement its main functionality:

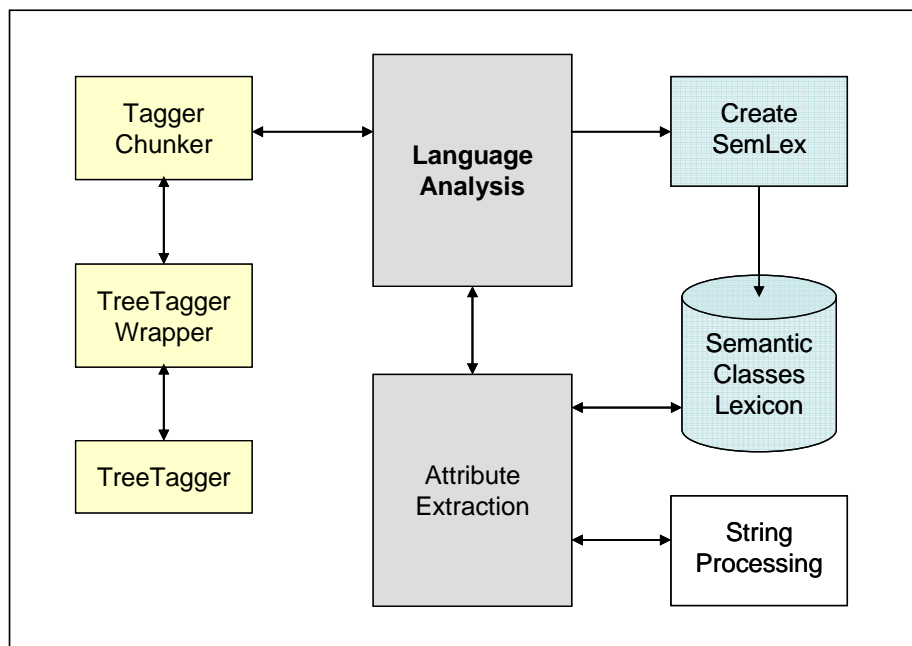


Figure 5.3: Interaction between LA classes and external and internal resources

The **Language Analysis class** is the LA main class and is responsible for the creation of the final interest nodes. It is also the interface between LA and DM. The Language Analysis class interacts with several other LA classes. It calls the **Tagger Chunker class** for syntactically analyzing the user answer and receives its result, a tagged and chunked string of user interests. For this purpose, the Tagger Chunker class accesses the external tool TreeTagger and the chunker by means of the **TreeTagger Wrapper class**. The Language Analysis class also calls the **Attribute Extraction class** which implements the nine Attribute Extraction Rules. The **String Processing class** has been developed in order to support the Attribute Extraction class with respect to various string manipulation tasks.⁷ For looking up the semantic classes of words, the Attribute Extraction class can access the **Semantic Classes Lexicon** stored in a database. The Semantic Classes Lexicon has been created semi-automatically by means of the **Create SemLex class**, which implements the functions needed for extracting wordlists from GermaNet and inserting them into a database. In the following sections, we describe how these classes interact for accomplishing the three main tasks of the LA.

⁷ Typical tasks it accomplishes are splitting the chunker output into phrases, returning the length of words or tags, separating tags from words, etc.

5.1.2.2 The Linguistic Analysis of User Answers

Attribute values cannot be extracted directly from raw user input. The user input has to be linguistically analyzed first, i.e. the input has to be annotated with linguistic labels on word and phrase level. These labels are needed for drawing inferences about the function of words and phrases and their suitability to serve as attribute values.

For analyzing user input linguistically, we use the TreeTagger [Schmid 1994], as introduced in Sections 3.3.2 and 3.4.2. It accomplishes the part-of-speech tagging, is associated with a chunker for shallow syntactic parsing, and performs sufficiently well for our input. The part-of-speech and chunk tag annotation is detailed enough to serve as a basis for the attribute value extraction. Tagging and chunking errors can either be captured by the Attribute Extraction Rules or are marginal enough to be neglected. As explained in Section 3.3.2, this was an important criterion for choosing this tool, because we do not yet have enough data at our disposal to train a statistical tool on our own data.

For analyzing a user answer linguistically, the LA main class calls the Tagger Chunker class, which provides access to the external TreeTagger tool. The LA passes the complete user answer (that may consist of several interests) to the Tagger Chunker class. For calling the TreeTagger, this class transforms each user interest into an array of individual words, because this is the input format the TreeTagger expects. The array of words is sent to the TreeTagger by means of the wrapper class (cf. Figure 5.3 above) which returns the tagged and chunked output. A wrapper is necessary here, because the TreeTagger is a compiled program that does not provide a JAVA-API (cf. Section 6.2.3.1). The Tagger Chunker class collects all tagged and chunked interests and sends them back together to the LA main class.

The TreeTagger adds part-of-speech tags to each word contained in the user interest (cf. Section 3.3.2) and the associated chunker finds syntactic units (chunks) (cf. Section 3.4.2). The part-of-speech tags used are based on the Stuttgart Tübingen Tagset [Schiller et al. 1995] (cf. Section 3.3.1). They are added right behind each word (e.g. “Bücher/NN”, “spannende/ADJA”). The chunk tags used for surrounding syntactic units are the following:

- <NC> ... </NC> nominal chunk (e.g. <NC> Autos (cars) </NC>)
- <VC> ... </VC> verbal chunk (e.g. <VC> lesen (reading) </VC>)
- <PC> ... </PC> prepositional chunk (e.g. <PC> im Zug (on the train) </PC>)

The user interest “spannende Bücher lesen im Garten” would be returned by the TreeTagger and the chunker in following form (tagged and chunked):

Example 5.5

```
<NC>
  spannende/ADJA      (exciting)
  Bücher/NN          (books)
</NC>
<VC>
  lesen/VVFIN        (reading)
</VC>
<PC>
  im/APPRART         (in the)
  <NC>
  Garten/NN          (garden)
  </NC>
</PC>
```

5.1.2.3 The Extraction of Attribute Values from the User Answer

In the previous section, we explained how a natural language user interest is analyzed linguistically by means of the TreeTagger and the chunker. The next step is to extract attribute value pairs from the tagged and chunked user interest. This task is accomplished by the Attribute Extraction class as described in the following.

For each user interest, the main method of this class creates a **vector of chunks** (i.e. syntactic units of the interest), i.e. a complete vector always represents a single user interest. For the interest “spannende Bücher lesen im Garten” in Example 5.5 above, this vector has three elements (cf. Example 5.6):

Example 5.6

vector of chunks = < <NC>spannende/ADJA Bücher/NN</NC>,
<VC>lesen/VVFIN</VC>,
<PC>im/APPRART<NC>Garten/NN</NC></PC> >

This vector of chunks, together with the original (unprocessed) user answer, represents the basis on which the Attribute Extraction class operates. It is used as input for the Attribute Extraction Rules (see below).

The class returns to the LA main method a **vector of attribute-value pairs**. In these pairs, the attributes are instantiated with the values that have been found in the user interest. In detail, this vector of attribute-value pairs contains elements of different types:

1. The first element is the **vector of objects**, i.e. a vector consisting of one or several `object` attribute-value pairs. Object attributes are collected as a separate vector because their number determines the number of interest nodes created from a single user interest (typically one, but several in case of coordinations, cf. Section 5.1.2.4).
2. Further elements are **other attribute-value pairs**, in case there are any.

For Example 5.6 above, the vector of attribute-value pairs would contain four elements, the first of which is the vector of objects (cf. Example 5.7). Here, the vector of objects contains only one element (which is the case for most interests). Besides the `object`, the vector in Example 5.7 contains three “other” attribute-value pairs.:

Example 5.7

vector of attribute-value pairs = < vector of objects = <object = Bücher>,
activity = lesen,
location = im Garten,
property = spannende >

Attribute Extraction Rules

The core of the Attribute Extraction class is the set of **Attribute Extraction Rules**. We developed these rules in order to extract specific attribute values needed for our interest nodes. They make use of syntactic, lexical, semantic, and heuristic criteria to determine suitable candidates for attribute values. We have implemented nine rules each that are specialized for the recognition of specific attributes. With them, we can handle most linguistic phenomena occurring in our user input. The rules show different degrees of complexity, because some attrib-

utes are more difficult to find in the user interest than others. For finding attribute values, the rules take into account syntactic categories of words, word and phrase order, lexical realizations of prepositions (i.e. the preposition “aus” may be treated differently than the preposition “von”), capitalization of words, etc. Some errors produced by the TreeTagger can even be captured by mere heuristic observations integrated into these rules.

For the attribute extraction, some words in the user input serve as semantic keywords. This means that they carry a specific semantics, which determines whether they can be part of a particular attribute value. “München (Munich)”, for example, has the semantics of being a location. Therefore, the embedding phrase (e.g. “im schönen München (in beautiful Munich)”) can serve as value for the attribute `location`. The problem we had to solve for this purpose was to determine which words can serve as semantic keywords and to recognize their semantics automatically. This was the reason for developing and employing the Semantic Classes Lexicon.

The Attribute Extraction Rules are processed sequentially in a fixed order until all attribute values have been extracted, if possible. For each attribute value found, an attribute-value pair is created and added to the vector of attribute-value pairs. In the following, we present the nine rules in the order in which they are called. Afterwards, we discuss the role of this order and give an example for the application of the rules.

- **Rule 1:** extracts value for attribute `activity`

It collects all verbs (including auxiliaries) from the vector of chunks and stores them as the `activity` value. In most cases, only a single main verb occurs in a user interest. For finding verbs, it compares the part-of-speech tags in the user interest with all verb tags in the STTS tagset (cf. Section 3.3.1). It explicitly avoids participle verbs, however, occurring within noun phrases (e.g. “gelesene” in “das gelesene Buch (the book read)”).

- **Rule 2:** extracts value for attribute `frequency`

All elements of the vector of chunks (except verb phrases) are checked: Does any of them contain a word of the semantic class “temporal expression” (i.e. which is listed with label `TEMP` in the Semantic Classes Lexicon)? If yes, the entire phrase surrounding this word is taken as value of `frequency`. All such temporal phrases found are added to the value.

- **Rule 3:** extracts value for attribute `location`

All elements of the vector of chunks (except verb phrases) are checked: Does any of them contain a word of the semantic class “location” (i.e. which is listed with label `LOC` in the Semantic Classes Lexicon)? If yes, the entire phrase surrounding this word is taken as value of `location`. All such locational phrases found are added to the value.

- **Rule 4:** extracts value for attribute `participants`

For this attribute, the extraction happens in two steps:

1) Prepositional phrases in the vector of chunks which contain a preposition “mit” or “zu” are checked: Does any of them contain a word of the semantic class “human being” (i.e. which is listed with label `HUM` in the Semantic Classes Lexicon)? If yes, the entire phrase surrounding this word is taken as the value of `participants`. All such phrases found that denote human beings are added to the value.

The restriction on “mit”- and “zu”-phrases occurs from the heuristic observation that these are typical cases where participants occur in prepositional phrases (e.g. “mit Freunden (with friends)”, “zu zweit (twosome)”).

2) In the second step, we have to check all remaining phrases of the vector of chunks for keywords indicating a participation mode (e.g. “alleine (alone)” in “alleine faulzen (being lazy alone)”: Does any of the phrases contain a participation keyword (i.e. a word listed with label `PART` in the Semantic Classes Lexicon)? If yes, the entire phrase surrounding this word is taken as or added to the value of `participants`. All phrases containing a word indicating a participation mode are added to the value.

- **Rule 5:** extracts value for attribute `rating`

All elements of the vector of chunks are checked: Does any of them contain a word of the semantic class “rating” (i.e. which is listed with label `RATE` in the Semantic Classes Lexicon)? If yes, the entire phrase surrounding this word is taken as value of `rating`. All such rating phrases found are added to the value. Typical ratings are, for example, “besonders (particularly)” or “speziell (especially)”.

- **Rule 6:** extracts value for attribute `dislike`

All elements of the vector of chunks are checked: Does any of them contain a word of the semantic class “dislike” (i.e. which is listed with label `DIS` in the Semantic Classes Lexicon)? If yes, the entire phrase surrounding this word is taken as value of `dislike`. All such dislike phrases found are added to the value. Typical dislikes are, for example, “nicht (not)” or “keinesfalls (in no case)”.

- **Rule 7:** extracts value for attribute `favorite_items`

This rule operates on the original user answer. Each word in the answer is checked as to whether it begins with a capital letter. Two or more words **in a row** which begin with a capital letter are added to the `favorite_items` value. Consequently, isolated words that begin with a capital letter (i.e. which are not in a row) are ignored by this rule. All such units consisting of two or more words with a capital letter found are added to the `favorite_items` value. Finally, all `favorite_items` values found in the original user answer are removed from the vector of chunks. This is important, because Rule 8 may only operate on a vector of chunks from which favorite items have already been removed.

- **Rule 8:** extracts value for attribute `object` and attribute `nominal_modifier`; triggers extraction of value for attribute `property` (**Rule 9**)

This rule finds an `object` value and also possible properties and nominal modifiers⁸ related to this object.

In order to find an object, all noun phrases contained in the vector of chunks are checked: Does any of them contain words with part-of-speech tag `NN` (noun) or `NE` (proper name)? The first word with such a tag is taken as value of `object`. In case, several such words are found (which is normally only the case for coordinations, cf. Section 5.1.2.4), each of them constitutes the value of an individual `object` attribute.

⁸ Nominal modifiers are optional or necessary supplements of the noun, which express part of the noun meaning. An example for an optional nominal modifier is “aus Plüsch (plush)” in “Sofas aus Plüsch (plush sofas)”.

This is because separate interest nodes have to be created for each object (cf. Section 5.1.2.4). Since `object` is an obligatory attribute, its value is “–” in case no object is found at all in the user interest (cf. Section 4.4.2.1).

For each object found, **Rule 9** is called in order to find possible `property` attributes for the object (see **Rule 9**).

The next step is to detect a modifier following the noun that can serve as value of the attribute `nominal_modifier`. In the current state of the approach, only the first post-nominal modifier is found. This proved sufficient for our input data, which did not show examples of more than one post-nominal modifier. If desired, the rule can be extended to capture further modifiers, however.

- **Rule 9:** extracts value for attribute `property`

This rule is called from within **Rule 8** for a specific object. For each word of the noun phrase that surrounds the value of this object, the following is checked: Does this noun phrase contain words that have an ADJA (pre-nominal adjective), CARD (cardinal number), or VVPP (participle) tag? If yes, all words in the phrase with these tags are taken as `property` value. Moreover, all adverbial modifiers (e.g. “sehr (very)” in “sehr guter Wein (very good wine)”) of words with ADJA, CARD, or VVPP tag are collected and also stored in the `property` value.

In general, the **order** in which the rules are applied is arbitrary, except for Rules 7, 8, and 9. Rule 9 is called from within Rule 8, because it handles a specific aspect of the noun phrase surrounding the object. Further, it is crucial that Rule 7 is applied before Rule 8, since the words that have been used as value by Rule 7 are removed from the vector of chunks in order to avoid that they are reused by Rule 8. This means that `favorite_items` have to be extracted before `objects` for the following reason. The general idea is that the `favorite_items` value captures proper names in the user interest, whereas the `object` value captures nouns. This idea could not be realized easily, however, because we had to circumnavigate **tagging errors that occur for proper names**. In the approach we took, favorite items are complex nouns (part-of-speech tag NN) or complex proper names (NE). “Complex” means that they consist of at least two word units (e.g. “Harry Potter”). If favorite items are extracted before the object is extracted and are removed from the vector of chunks, this guarantees that only single-unit nouns (e.g. “Bücher (books)”) are left in the vector and used as values for objects. We are aware that with this approach we cannot capture favorite items that are single-unit proper names (e.g. “Fiat”, “Japan”). These can be used as `object` values instead. We had to take this approach of “two-or-more units” favorite items, however, in order to handle tagging errors for proper names. As outlined in Section 3.3.2, taggers often have problems with the recognition of proper names, i.e. they fail to assign the tag NE correctly. If we only relied on the tagger output and collected all words tagged with NE as favorite items, we could not capture many cases that were erroneously assigned other tags instead of NE. By collecting all capital-letter nouns that occur in a row we can at least guarantee capturing correctly all complex proper names in the German language. With this solution, we were able to improve the number of extracted `favorite_items` significantly.

We illustrate the functioning of the Attribute Extraction Rules in Example 5.8, which shows a very complex user interest, such that many rules will actually find attribute values:

Example 5.8

original user answer =

“am Wochenende spannende Bücher lesen im Garten, Harry Potter besonders”

(at the weekend reading exciting books in the garden, Harry Potter especially)

```
vector of chunks = < <PC>am/APPRART<NC>Wochenende/NN</NC></PC>,
                    <NC>spannende/ADJA Bücher/NN</NC>,
                    <VC>lesen/VVFIN</VC>,
                    <PC>im/APPRART<NC>Garten/NN</NC></PC> ,
                    ,/$,
                    <NC>Harry/NE Potter/NN</NC>,
                    besonders/ADV >
```

Note, that for Rules 1 – 6 and 9 the attribute values found are *not* removed from the vector of chunks. The removal is only accomplished for Rule 7 (`favorite_items`) for the reasons explained above. In Example 5.8, first, **Rule 1** is applied and extracts “`activity = lesen (reading)`”, because “lesen” has a verb part-of-speech tag (VVFIN). Next, **Rule 2** finds the temporal expression “Wochenende” and hence creates the attribute-value pair “`frequency = am Wochenende (at the weekend)`”. **Rule 3** finds the word “Garten” of the semantic class of locations and creates “`location = im Garten (in the garden)`”. **Rule 4** finds no words in the user input that are listed in the Semantic Classes Lexicon as either HUM or PART, hence the `participants` attribute is not set. **Rule 5** identifies “besonders” as belonging to the semantic class of ratings and sets “`rating = besonders (especially)`”⁹. Since the answer denotes an interest of the user, not a dislike, no word belonging to the semantic class “dislike” can be found by **Rule 6** and the `dislike` attribute is not set. **Rule 7** searches in the original answer string for two or more words in a row beginning with a capital letter and creates “`favorite_items = Harry Potter`”. In this case, the comma between “Garten (garden)” and “Harry” avoids that “Garten” is also added to this value. For spoken input, it might therefore be helpful to pronounce the “comma” explicitly. It is important to note that the noun phrase `<NC>Harry/NE Potter/NN</NC>` has now been removed from the vector of chunks. Finally, the most complex rule is applied: **Rule 8** finds the remaining noun phrase `<NC>spannende/ADJA Bücher/NN</NC>` and extracts from it “`object = Bücher (books)`”. It calls **Rule 9** for this noun phrase, which finds a property related to this object: “`property = spannende (exciting)`”. Finally, **Rule 8** could not find a post-nominal modifier and the attribute `nominal_modifier` is not set. The attribute extraction task is finished.

Note that the formulation of the Attribute Extraction Rules might lead to overgeneration for longer or more complex user input. For our restricted input, however, which scarcely shows more complexity than in the above example, they yield good results.

5.1.2.4 The Creation of Interest Nodes

After the attribute values have been extracted as described in the previous section, they are returned to the LA main method as a vector of attribute-value pairs. One such vector is returned for each user interest. From each vector, the LA main method creates one or several interest nodes that will be inserted into the profile later by the PM.

⁹ At the moment, the rating value consists of the entire phrase surrounding the rating keyword. In Example 5.8, however, the rating keyword occurs as an isolated element in the vector of chunks and is hence stored as the only value of the `rating` attribute. A solution for such cases might be to add the preceding noun phrase to the `rating` value too.

Typically, exactly one interest node is created from one vector of attribute-value pairs. Since, however, we also wanted to set the technical basis for handling coordinations (e.g. “Emails und Zeitungen (emails and newspapers)”) with our approach, we included the possibility that two or more interest nodes can be created from a single vector of attribute-value pairs representing a coordination (see below).

An interest node is created as follows. An XML representation of an interest node is constructed by adding XML tags surrounding the node (<interest></interest>) for the first element in the vector of objects (e.g. “Emails”). The created interest node contains this object attribute-value pair (e.g. “object = Emails”) and all further attribute-value pairs contained in the vector of attribute-value pairs (e.g. “activity = lesen (reading)”, “frequency = ständig (all the time)”). In the case that the vector of objects contains another object (e.g. “Zeitungen”), a new XML interest node is created for this object, containing the object attribute-value pair (“object = Zeitungen”) and all further attribute-value pairs contained in the vector of attribute-value pairs, which have already been used together with the first object (e.g. “activity = lesen (reading)”, “frequency = ständig (all the time)”). A separated XML node is created for each object contained in the vector of objects according to this schema.

Figure 5.4 shows two sample interest nodes in XML. We also indicate the TreeTagger and chunker output for a better understanding of how the attribute values were derived:

TreeTagger and chunker output 1:

speziell (especially)/ADJD<NC>orientalische (oriental)/ADJA Länder (countries)/NN<NC>

Created XML interest node 1:

<interest object="Länder" property="orientalische" rating="speziell"> </interest>

TreeTagger and chunker output 2:

<VC>wandern (hiking)/VVFIN</VC><PC>in (in)/APPR<NC>den (the)/ART Alpen (Alps)/NN<NC></PC><PC>mit (with)/APPR<NC> meinen (my)/PPOSAT Kindern (kids)/NN<NC></PC>

Created XML interest node 2:

<interest object="-" activity="wandern" location="in den Alpen" participants="mit meinen Kindern"></interest>

Figure 5.4: Language Analysis output examples

Finally, all interest nodes created for a user answer are handed over to the DM, which passes them on to the PM for locating them in the profile graph (cf. Section 5.2).

The Coordination Problem

In the empirical user study described in Section 6.1, users occasionally also entered coordinations of interests (e.g. “Bücher und Zeitungen (books and newspapers)”). *Coordinations* are two or more phrases (*coordinate phrases*) of the same syntactic type that are separated with comma, “und (and)”, “oder (or)”, or similar *conjunctions*. Being of “the same syntactic type” means, for instance, that all are noun phrases or all are verb phrases, etc.

Simple coordinations such as “Bücher and Zeitungen” are not problematic in our approach. A different interest node can be created for “Bücher” and for “Zeitungen”. Coordinations become problematic, however, if further words co-occur, as for instance “öfters (often)” and “lesen (reading)” in Example 5.9:

Example 5.9

“öfters Bücher und Zeitungen lesen (often reading books and newspapers)”
→ vector of objects = [object1 = Bücher, object2 = Zeitungen]
other attributes = [frequency = öfters, activity = lesen]

The LA is already technically capable of creating different interest nodes for coordinations. This is achieved by its ability to process not only one object per user interest, but *several* objects with distinct values (e.g. “object1 = Bücher”; “object2 = Zeitungen”). Each of these object attribute-value pairs is stored in the vector of objects (cf. Section 5.1.2.3). The vector of objects is in turn the first element of the vector of attribute-value pairs. However, the LA is not yet capable of disambiguating the exact references of the other attributes that occur in a coordination. For the interest shown in Example 5.9, the vector of objects contains two objects. As a consequence, two interest nodes are created, one for “Bücher” and one for “Zeitungen” (cf. Example 5.10). All other attributes are added to both nodes, i.e. the attributes *frequency* and *activity* are assumed to refer to both objects equally and no disambiguation takes place:

Example 5.10

Node 1: <interest object=“Bücher” activity=“lesen” frequency=“öfters”></interest>
Node 2: <interest object=“Zeitungen” activity=“lesen” frequency=“öfters”></interest>

In fact, it is possible that “öfters (often)” only refers to reading books, but not to reading newspapers. This problem may get even more complicated with more complex user interests.

For this reason we decided not to allow coordinations as user input at the present state of the system. Coordination handling is very difficult from a linguistic point of view and we did not intend to elaborate it fully within the scope of this thesis. We have, however, already implemented the technical prerequisites to handle this phenomenon. For realizing a complete coordination handling, one would have to add nontrivial linguistic methods for determining the correct reference of words co-occurring in the coordination and for assigning further attributes correctly to the different interest nodes.

5.2 The Profile Manager

In this section, we describe the **Profile Manager (PM)**. Together with the Profile Model (cf. Section 4.4), it constitutes the main component of the *profile maintenance* part of our system. This section is structured as follows. Section 5.2.1 gives an overview of the PM, its tasks, and embedding in the profiling system. In Section 5.2.2, the functioning of the PM component is discussed in detail. We first describe the PM classes. The remaining two sections describe the main tasks of the PM: the initial construction of a profile for a new user and the process of augmenting and updating an existing profile during the refining dialog.

5.2.1 Overview of the Profile Manager: Tasks and Embedding

After the LA has created a set of interest nodes, the DM adds to this set of nodes the information as to whether it contains information for creating a new profile (for a new user) or for updating an existing profile. This distinction is made by means of the **profile ID**, a unique number assigned to each profile by the PM when creating it. While the PM is responsible for creating profile IDs, the DM is responsible for keeping track of them and controlling the PM actions that depend on the existence of a profile ID. The DM either adds an existing or an

empty profile ID to the set of nodes it passes to the PM. The PM then interprets the value of the profile ID in order to choose between two possible actions:

1. In case the **profile ID is empty**, the task of the PM is to **create a new profile** from the set of interest nodes it receives from the DM. It creates a unique profile ID for this new profile.
2. In case the **profile ID already exists** in the profile database, the task of the PM is to **update the profile** with this ID, i.e. to add or remove information according to the interest nodes it receives from the DM.

After the PM has created or updated the profile, it returns it to the DM together with its profile ID. Figure 5.5 shows the interface between PM and DM:

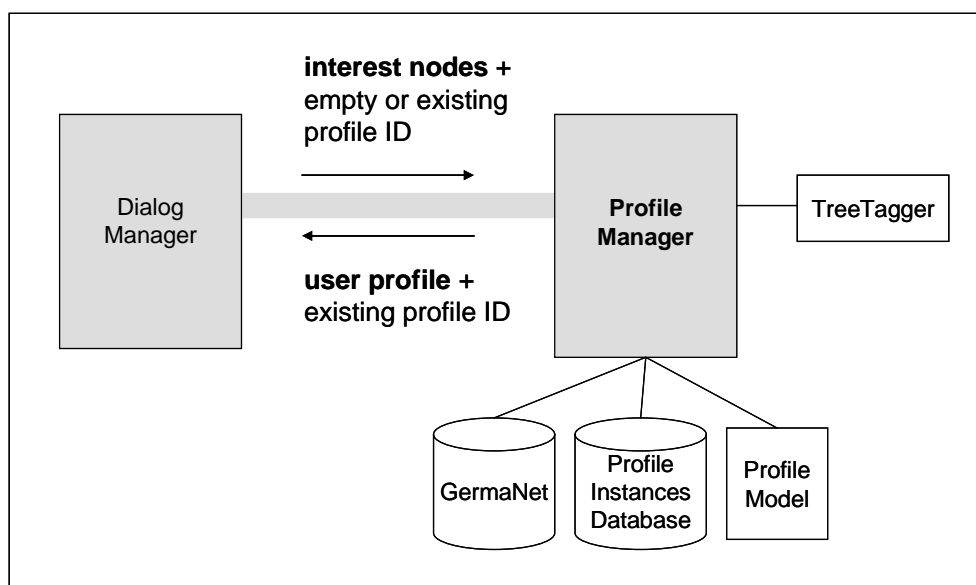


Figure 5.5: Interaction between Profile Manager and Dialog Manager

The PM creates user profiles according to the **Profile Model**. The Profile Model defines the structures and contents of profile instances (cf. Section 4.4). The profiles created for individual users are stored in the **Profile Instances Database**, an SQL database located on a central server. User profiles constitute a semantic graph representation of user interests. In order to find semantic relations between user interests or to locate new interests in the profile graph, the PM makes use of the lexical-semantic resource **GermaNet** (cf. Section 3.5.1). Like the LA, the PM also has access to the **TreeTagger**, which it uses for finding the base forms (e.g. “Buch (book)”) of morphologically inflected words (e.g. “Bücher (books)”). This information is needed for finding user interests in GermaNet, which only contains base forms.

5.2.2 Functioning of the Profile Manager

In this Section, we first describe the classes of the PM and their interaction. We will then explain the two main tasks of the PM, creating and updating user profiles, in more detail.

5.2.2.1 Profile Manager Classes

The most important classes of the PM that realize its main functionality are shown in Figure 5.6. Details that are only relevant for the implementation and system performance, will not be discussed here, but in Chapter 6.

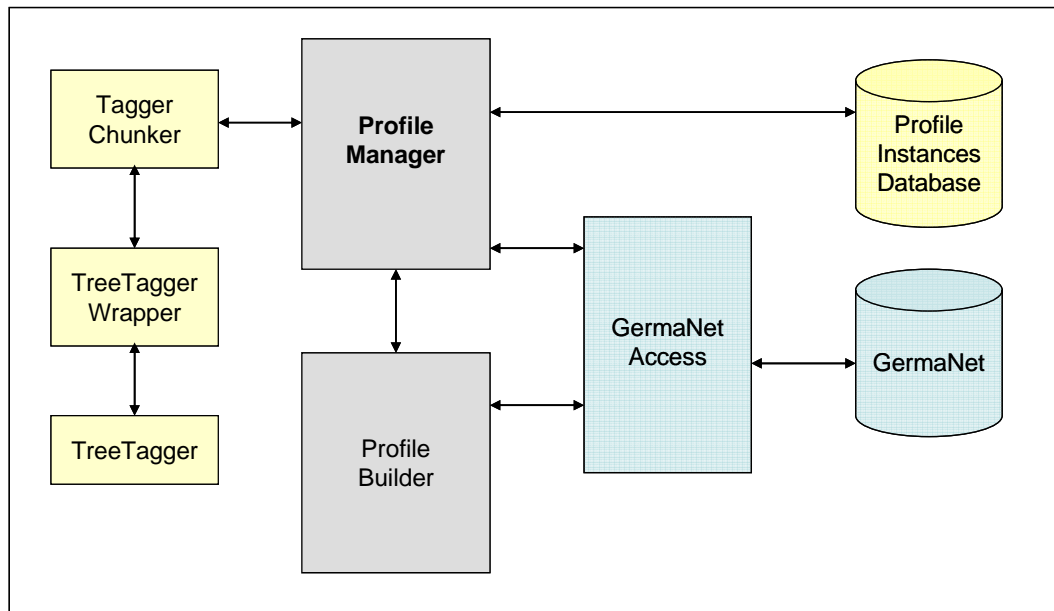


Figure 5.6: Interaction between PM classes, GermaNet, TreeTagger, and Profile Instances Database

The main class is the **Profile Manager class**, which is also the interface to the DM. It is responsible for storing profiles in the **Profile Instances Database** and reading them from there. Its core task is to locate interest nodes in GermaNet. It does so by searching in a fixed order for the `object`, `favorite_items`, or `activities` contained in an interest node in GermaNet according to a specific algorithm. For this purpose it first converts the attribute values into their base forms by means of the lemmatizer function of the **TreeTagger**. It accesses the TreeTagger by means of the **Tagger Chunker class** and **TreeTagger Wrapper class** (cf. Section 5.1.2.1). The Profile Manager class also determines what happens with interest nodes that cannot be found in GermaNet.

The **Profile Builder class** supports the Profile Manager class by manipulating the profile graph. It inserts paths and nodes and also newly acquired attribute-value pairs or attribute values. It transforms the original GermaNet-internal data format into the profile graph format of LINGUINI. For this purpose, it uses the GermaNet data structures provided by the GermaNet Access class. The **GermaNet Access class** functions as an API to the lexical-semantic resource **GermaNet**. The information contained in GermaNet is represented by a large number of XML files that are read by the GermaNet Access class, which offers a set of access methods and data structures for this information.

In the following, we describe in more detail, how the PM accomplishes its two main tasks, the construction of a new profile and the update of an existing profile.

5.2.2.2 Constructing a Profile Graph

In this section, we describe the first main task of the PM, the initial construction of a profile for a user who communicates with the system for the first time. In Section 5.2.1, we mentioned that the DM calls the PM with a set of interest nodes and a profile ID, which may be either empty or not. If the profile ID is empty, this is the signal for the PM to create a new profile.

In order to create a new profile, the **Profile Manager class** first reads the input it has received by the DM: one or several interest nodes created by the LA. It then tries to construct a profile graph from these interest nodes. In order to semantically relate them to each other and to find superordinate nodes from which it can build the graph structure, it looks up the attribute values which determine the names of the interest nodes in GermaNet according to the search algorithm shown in Figure 5.7 below.

Before we explain the search algorithm, a remark about **synonym recognition** should be made. If users are allowed to use their own words instead of choosing from a set of predefined values only, we have to make sure that synonyms can be recognized. *Synonyms* are different words with the same meaning (e.g. “Auto” and “PKW” both have the meaning “car”). It has to be checked, whether a word entered by the user has the same meaning as a word existing in the Target Group Model or GermaNet. Since GermaNet (and hence the Target Group Model derived from it) stores synonym sets (i.e. all synonyms representing a specific meaning), we can easily determine whether two different words refer to the same meaning. For this to be the case, they both have to be members of the same synonym set. If a word entered by the user is in the same synonym set as a target group node name, for example, we can infer that they mean in fact the same, even if the words themselves differ.

Before an attribute value contained in an interest node can be searched for in GermaNet, it has to be transformed into its base form, because GermaNet does not contain morphologically inflected words. For this task the **lemmatizer function of the TreeTagger** is used, which has been described in Section 3.3.2. The TreeTagger receives an inflected word (e.g. “Bücher”) and returns the corresponding base form (e.g. “Buch”). After this transformation of the attribute values in an interest node, the search algorithm can be applied (cf. Figure 5.7):

1. If an **object value is set** in the interest node, check if it occurs in GermaNet. If yes, insert all paths found in GermaNet for this value into the profile and stop. If no, continue with step 2.
2. If no object value but a **favorite_items value is set**, check if it occurs in GermaNet. If yes, insert all paths found in GermaNet for this value into the profile and stop. If no, continue with step 3.
3. If no object and favorite_items but an **activity value is set**, check if it occurs in GermaNet. If yes, insert all paths found in GermaNet for this value into the profile and stop. If no, insert the interest node under the “NOT FOUND node” and stop.

Figure 5.7: Search algorithm for locating interest nodes in GermaNet

Note that the search process operates on the **XML tree structure** into which we have transformed the original GermaNet graph (cf. Section 4.4.2.2). The PM takes all paths it finds in GermaNet between an interest node and the GermaNet root node and inserts them into the user profile. A disambiguation, i.e. the selection of a path that represents a specific meaning and the removal of other paths (i.e. other meanings), is not yet integrated into our system, but

could theoretically be inserted in the processing steps of the DM as described in Section 5.4.2.2.

The three attributes `object`, `favorite_items`, and `activity` have been chosen for this search algorithm, because we consider them the most important information-carrying elements of an interest node. Most user interests that occurred in our user studies were formulated as nouns, proper names, or verbs, which provide exactly the values for these three attributes. The other attributes merely represent additional information, which typically does not occur alone in a user interest. Our study about preferred user input (cf. Section 6.1) has shown that at least one of the attributes `object`, `favorite_items`, or `activity` is always present in a user interest. This was an important requirement for selecting them for the search algorithm, because it ensures a consistent profile construction for all interest nodes. The other attributes do not fulfill this requirement.

The **order** we chose for the three attributes in the search algorithm expresses our view that the object is the most important semantic element of an interest node, if present, followed by a favorite item, if no object is given. Activities that are instantiated with a transitive verb (e.g. “trinken” in “Kaffee trinken (drinking coffee)”) or a ditransitive verb (e.g. “legen” in “Zeitschriften ins Regal legen (putting magazines on the shelf)”), for example, often do not reveal as much user-individual information as the objects they refer to (“coffee” and “magazines” in this example). Therefore, they occur only at position three in our algorithm. Many users like the activity “reading”, for example, but what is more interesting for a personalization service is the types of objects they read (e.g. magazines, books, online articles, etc.). We found that this observation holds for the majority of cases that occurred in our studies. Theoretically, a different order of the three attributes is possible and the algorithm can be adapted accordingly, if this is desired.

It is furthermore important to note that in this algorithm all attribute values are searched in the **noun part** of GermaNet. In Section 4.4.2.3, we have explained why this is possible even for activities (i.e. verbs) and that we consider this necessary in order to keep the graph structure homogeneous.

In case an interest node cannot be found in GermaNet with the algorithm in Figure 5.7 above, it is inserted under the “**NOT FOUND node**”. The “NOT FOUND” node subsumes all interests that could not be found in GermaNet and could hence not be inserted in the profile graph at a meaningful position. We will see in Section 5.4.2.2 that the DM tries to relocate these nodes automatically by exploiting specific information from the Target Group Model. The NOT FOUND node does not necessarily contain any child nodes. In case all interests of a specific user could be found in GermaNet, the NOT FOUND node in her profile simply stays empty.

The user profile is created on the basis of the Profile Model described in Section 4.4, which determines the structure of profile graph and profile nodes. Since a new profile is constructed from the **starting answer** of the user, the number of its profile nodes depends on the (configurable) number of interests requested for the starting answer. In our final user study for evaluating LINGUINI (cf. Section 6.3), we requested two to three interests in the starting answer. As a consequence, the initial profile graph is rather “slim” and the major part of the information is acquired only later, during the refining dialog. If for some reason a rather short dialog is preferred for a specific application, it is possible to acquire more information in the starting answer, thus creating a rich profile in this first step, and then to ask less questions in the refining dialog. This is possible because the number of interest nodes used for the initial creation of a profile is technically not restricted in our system. In the next section, we describe how this initial profile can be augmented with further information acquired during the dialog.

5.2.2.3 Augmenting and Updating the Profile

In this section we describe how an existing user profile can be augmented or updated with new information, henceforth called *updating process*. This is typically what happens during the refining dialog. The refining dialog starts with the next question following the starting answer of the user. Until the beginning of the refining dialog, the initial profile has been constructed from the starting answer and the TGMAT has been performed afterwards. From the Matching Result, the DM has selected the node pair about which the next question is asked (cf. the overview of this process in Section 4.2.2.2).

The profile node in this selected node pair has a specific **node ID**. This node ID is important for the PM, because it determines the place in the profile graph where the new information has to be inserted. Information extracted from the answer to the question about this node pair is supposed to be inserted into the profile node with this ID.

The updating process is initiated by the DM. The DM passes a **set of interest nodes**, which represents the most recent user answer, to the PM. It also passes an **existing profile ID** to the PM, which indicates which user profile is supposed to be updated. For the updating process, the DM also sets the boolean parameter **isUpdateInput**, which tells the PM exactly which type of update it is supposed to perform on the existing profile. The difference between the two possible update types is whether an existing profile node should be updated or a new node should be inserted into the profile:

1. In case **isUpdateInput is true**, the task of the PM is to augment an **existing profile node** with the newly acquired information. In this case, the DM also passes the above mentioned node ID for the node which is to be updated.
2. In case **isUpdateInput is false**, the task of the PM is to create a **new profile node** from the newly acquired information. The DM does not need to pass a specific node ID here.

Update Type 1 refers to cases in which a **deepening question** (introduced in Section 4.2.1.2) about a specific interest has been asked by the DM and the answer is now passed to the PM. Update Type 2, on the other hand, refers to the case in which a **broadening question** has been asked.

The Profile Manager class loads the profile with the given profile ID from the Profile Instances Database, performs the changes, stores the profile and always returns the complete profile to the DM. In the following, we explain in more detail how the two update types are accomplished.

Update Type 1: Augmenting an Existing Profile Node

For this type of update, the profile node with the indicated node ID is first located in the profile. Next, the attribute-value pairs of this profile node are compared with the new information received in form of an interest node by the DM. In case the attribute-value pairs differ in their values, the values in the profile node are updated. The PM then checks whether there are further attributes in the new user input which do not yet exist in the profile node. These new attribute-value pairs are then added to the profile node. Hence, a profile node can be updated in two possible ways:

- A **new attribute-value pair** is added to the profile node.
- A **new attribute value** is added to an existing attribute, which has one or several values already.

We would like to illustrate these two actions by means of an example. Assume that the user profile already contains the following profile node (cf. Figure 5.8) representing the interest “Bücher lesen (reading books)”:

```
<Buch id="nArtefakt.2711.Buch"
  activity="lesen"
  object="Bücher"/>
```

Figure 5.8: Profile node “Bücher lesen (reading books)”

A **new attribute-value pair** is added, for example, if the user mentions that the books she reads have to be exciting. The interest node created from the user input is compared with the existing profile node. Since the existing node shown in Figure 5.8 does not yet have an attribute `property`, the attribute-value pair “`property = spannende (exciting)`” will be added to the node (Figure 5.9):

```
<Buch id="nArtefakt.2711.Buch"
  activity="lesen"
  object="Bücher"
  property="spannende"/>
```

Figure 5.9: A new attribute-value pair has been added

A **new attribute value** is added to some existing attribute, for example, if the user mentions during the dialog that she not only reads but also writes exciting books. The corresponding attribute (`activity`) already exists in the profile node. Hence, not a completely new pair, but only a new value, “`schreiben (writing)`”, is added to the `activity` attribute (Figure 5.10):

```
<Buch id="nArtefakt.2711.Buch"
  activity="lesen, schreiben"
  object="Bücher"
  property="spannende"/>
```

Figure 5.10: A new attribute value has been added

Note that we do not check semantic **contradictions between attribute values**, because this is very difficult and in many cases not possible at all without asking the user for help. We also do not **delete profile nodes** automatically from the profile, except for the case of relocating child nodes of the NOT FOUND node (see below). **Attribute-value pairs** are also not deleted in general, but their **values** can be overwritten with more up to date information. Users can, of course, change or delete information from their profiles manually anytime via a user interface to the profile.

Update Type 2: Creating a New Profile Node and Relocating NOT FOUND Nodes

The second type of update inserts a new profile node, which represents the new user interest passed to the PM. This is accomplished in the same manner as the construction of a new profile described in Section 5.2.2.2. The attributes of the interest node are searched for in GermaNet according to the search algorithm (cf. Figure 5.7 above). If none of them is found, the interest is located under the NOT FOUND node.

A special case should be mentioned here: the **relocation of child nodes of the NOT FOUND node**, henceforth shortly called “**relocation of NOT FOUND nodes**”. In this case, a

node which could previously not be located in GermaNet and is hence temporarily stored under the NOT_FOUND node, is enhanced with new user information. It can then, in some cases be relocated in the profile graph, i.e. inserted at the semantically correct position.

For the relocation of a NOT FOUND node, several steps are necessary. First, the node is updated with new information by either adding new attribute-value pairs or values as described in the previous section. Furthermore, a check has to be made as to whether the node can now (after the information update) be found in GermaNet according to the search algorithm. If it can be found, it is relocated in the profile and gets a **new node ID** (derived from GermaNet).

The assignment of a new node ID is due to the fact that NOT FOUND nodes have a different ID set than the rest of the graph. If a node can be relocated, its previous version under the NOT FOUND node is deleted. If it still cannot be found in GermaNet, the node stays under the NOT FOUND node. Since several questions can be asked about a node, it is possible that the relocation is tried several times for the same node during the dialog. Example 5.11 illustrates this process:

Example 5.11

A user mentions, for example, the interest “Goldenes Blatt” (a German yellow press magazine). The interest node, the only attribute-value pair of which is “favorite_items = Goldenes Blatt”, cannot be found in GermaNet and is hence stored under the NOT FOUND node. During the refining dialog, the system receives the information that this is a special type of magazine. It adds to the node the attribute-value pair “object = Zeitschrift (magazine)”. The augmented node is now <interest object = ”Zeitschrift”, favorite_items = ”Goldenes Blatt”></interest>. Then, it searches for the object value “Zeitschrift” in GermaNet (according to the search algorithm in Figure 5.7), finds it, and can now insert the corresponding GermaNet path into the profile and locate the augmented interest node correctly under the intermediate node “Druckwerk (print work)”.

5.3 Target Group Matching and Target Group Model

In this section, we discuss the **Target Group Matching (TGMAT)** component and the **Target Group Model**. The TGMAT contributes to the question selection process the DM has to accomplish (cf. Section 5.4). It is therefore an important prerequisite which we would like to discuss before turning to the functionality of the DM.

As in the previous sections, we first give an overview of the TGMAT in Section 5.3.1. We describe the main tasks and embedding of the TGMAT and introduce the Target Group Model, the central underlying model for the TGMAT.

In Section 5.3.2, the functioning of the TGMAT component is explained in detail. The TGMAT classes and main data structures are introduced and the TGMAT process is illustrated by means of the TGMAT main function. The remaining sections explain the tasks of the TGMAT in detail: the preparations that are necessary for the TGMAT and how the most similar target group is determined.

5.3.1 Overview of the Target Group Matching

5.3.1.1 Tasks and Embedding of the Target Group Matching

Before the DM selects the next question, it consults the TGMAT. The TGMAT compares the user profile with each **target group (TG)** in the Target Group Model and determines the TG that is most similar to the actual state of the profile.

Figure 5.11 below illustrates the interface between TGMAT and DM. The TGMAT receives three kinds of information from the DM:

- The actual **user profile**
- The **IDs of the TGs** contained in the Target Group Model:

Formally, **TGs** look very much like user profiles. Like user profiles, they are stored in an SQL database, the **Target Group Database**, with a unique **TG ID** each. Since arbitrary Target Group Models, created for different applications, can be used in LINGUINI, the IDs of the TGs used are configurable and their values are set in the configuration file. After the TG IDs have been set in the configuration file, they are read by the DM, which tells the TGMAT the IDs of the TGs it has to use for the current application.

- The **dislike parameter** (named after the `dislike` attribute, cf. Section 4.4.2.1):

The dislike parameter is set to “true”, if the TGs contain not only interests but also **dislikes** of the user. This parameter is necessary, in order to avoid the situation that the TGMAT compares an interest with a dislike merely on the basis of their attribute values and finds that they are identical without taking into account that they in fact mean the opposite of each other. If a Target Group Model is used which contains only interests (no dislikes), the value is set to “false”, because in this case, the TGMAT does not need to distinguish interests from dislikes. According to this parameter, the TGMAT adapts its functioning automatically to the type of Target Group Model used. This allows for a flexible adaptation of the system to models with or without dislikes. The handling of dislikes is explained in detail in Section 5.3.2.4 (under “Special Case: Dislikes”).

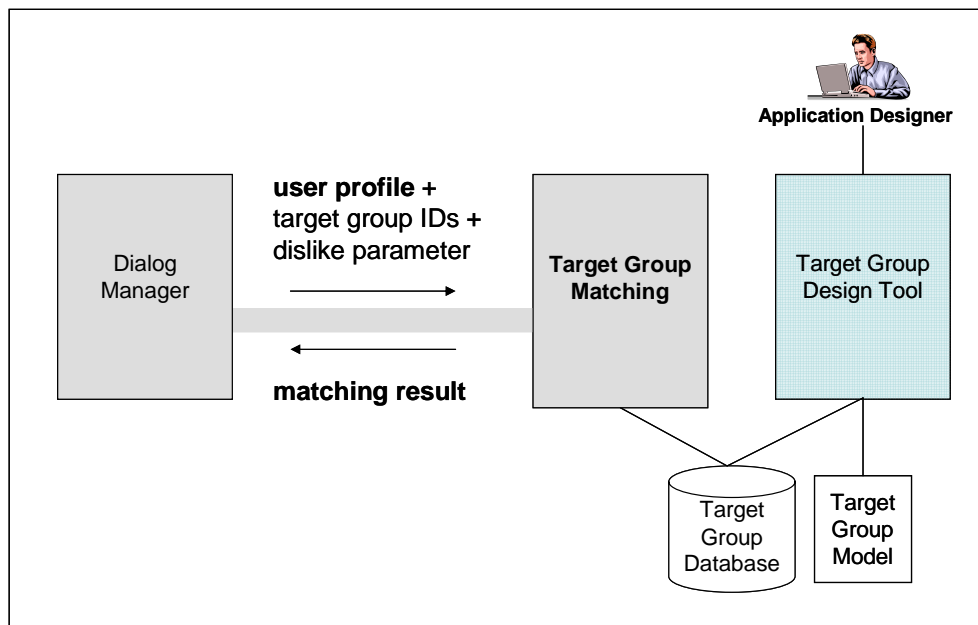


Figure 5.11: Interaction between Target Group Matching and Dialog Manager

The TGMAT returns to the DM the **Matching Result**, which contains the most similar TG(s). In case there are several TGs that all match the profile equally well, the Matching Result contains not only one, but all most similar TGs.

In order to determine the most similar TG(s), the TGMAT has to accomplish three tasks, which will be discussed in the remainder of Section 5.3:

1. Preparing a suitable information representation for user profile and TGs.
2. Comparing the profile with each TG in order to determine the most similar TG(s).
3. Creating the Matching Result.

At the bottom on the right in Figure 5.11, we can also see the **Target Group Model**. In our approach, different Target Group Models containing specific TGs can be created for each application separately if desired. For creating Target Group Models, a **Target Group Design Tool** can be applied (on the right in Figure 5.11), which has access to the “knowledge” of the Target Group Model schema (cf. Section 5.3.1.2). The created TGs are stored in the Target Group Database. It was not in the focus of this thesis to develop this Target Group Design Tool, however. Hence, we only sketch the underlying idea of it in the following paragraphs.

The typical users of the Target Group Design Tool are application designers, who have decided that they want to create a Target Group Model that is particularly characteristic for users of their application and which contains interests and attributes that are of particular relevance for them (e.g. interests strongly related to “sports” for a sports product recommender system). By creating a specific model, application designers can take considerable influence on the topics addressed and introduced by the system during the dialog (e.g. the dialog may centre on sports topics, if only these are represented in the Target Group Model).

The Target Group Design Tool should offer a user interface which allows selecting the nodes a TG should contain from GermaNet. The difficulty lies in the visualization of the complex graph structure of GermaNet and the different meanings of identical words (each represented by a different path). The tool has to support the application designer in selecting exactly the meaning she has intended for an interest node from GermaNet and in inserting it into the TG.

The tool could further integrate the functionality of creating a TG graph semi-automatically. For this purpose, interests characterizing a TG could be formulated as natural language phrases by the application designer. They would then be analyzed automatically and a graph would be created from them (similar to the creation of user profiles described in Section 5.2). After this, the application designer only has to eliminate superfluous GermaNet paths from the automatically created TG graph and keep the one that represents the intended meaning of a node. This semi-automatic process, with which we have created our Target Group Model, is further described in Section 6.2.3.3. This functionality of the tool could be realized easily, because the basis for it is already implemented by means of our LA and PM components. Merely a user interface and a possibility to visualize and remove paths from the graph would have to be developed. Until a Target Group Design Tool becomes available in the future, applications can use the specific Target Group Model we have created and which we currently use in LINGUINI. This model is described in the next section.

5.3.1.2 The Target Group Model

The **Target Group Model** can be considered a schema which defines how TGs can be structured. In this respect, it serves a similar function for TGs as the Profile Model for user profiles. The TGs created according to this schema for a specific application can be subsumed under the term **Target Group Model instance (TGMOD)**. With the term **TGMOD**, we also refer to the specific **Target Group Model instance we have developed for LINGUINI**. In cases where we refer to the underlying Target Group Model schema instead, we state this explicitly.

LINGUINI does not depend on a specific TGMOD or specific number of TGs, however. A TGMOD with an arbitrary number of TGs can be created for a particular application or the default instance we have created for our prototype system can be used. In this section, we explain this particular TGMOD we have developed for LINGUINI.

The Theoretical Basis of our TGMOD

The theoretical basis of our TGMOD is the sociological model developed by [Schulze 1992] (introduced in Sections 2.1.1.4 and 2.1.2.1). It contains 5 *milieus* (cf. Section 2.1.1.4), which represent different groups of the German population, and characterizes them by means of their lifestyle, interests, dislikes, and attitudes. The following is a brief summary of the main milieu characteristics (based on a more extensive description in [Alheit et al. 1994]):

- **Selbstverwirklichungsmilieu:** mainly students and “ascending educated elite”; scenes, cafés, and bars are central; open for new social “codes” and educational goals
- **Unterhaltungsmilieu:** mainly hedonists and young people later often belonging to other milieus; suspense is the predominant motivation
- **Niveaumilieu:** mainly intellectuals and “(haute)-bourgeoisie”; conservative thinking; importance of education, high quality, and style
- **Harmoniemilieu:** mainly proletariat and “lower middle class”; importance of correctness, modesty, and coziness; prefer inexpensive, simple things
- **Integrationsmilieu:** mainly “ascending middle class”; intersection between Niveaumilieu and Harmoniemilieu (cf. Figure 2.2 in Section 2.1.1.4); conformity is central

In Section 2.1.2.1, we have explained why we chose the approach of [Schulze 1992] for our TGMOD. The most important reasons were that it models both **interests and dislikes** and that these are represented in a **similar syntactic structure** to our user input, which makes the user input easily comparable to the information stored in the TGMOD.

In Schulze’s model, each of the 5 milieus is further characterized according to specific categories. For our TGMOD, we used only the interests and dislikes listed in Schulze’s category “**Alltagsästhetik (everyday life aesthetics)**”, because these showed the highest similarity to our typical user input. In spite of its rather abstract name, this category lists specific interests and dislikes formally comparable to those obtained in our user studies. We did not include other categories of Schulze that were less suitable for acquiring interest information, such as the personality of a typical group member or her overall living situation and attitudes.

In Schulze’s model, users are assigned to TGs according to a complex mechanism including various parameters. For assigning users to TGs in LINGUINI, the only Schulze parameter we use is **eta/Cramer’s V** (cf. [Schulze 1992]), which proved sufficient for our purposes. The parameter **eta/Cramer’s V** is assigned to each interest in a TG and expresses the discriminating power (“Diskriminationskraft” [Schulze 1992]) of the interest. The effect of this parameter is illustrated in Example 5.12:

Example 5.12

Assume that a TG contains the interest “Oper (opera)” which has eta/Cramer’s $V = .30$ (i.e. a high discriminating power) and the interest “Sammlungen (collections)” with eta/Cramer’s $V = .19$ (i.e. a lower discriminating power).

For a user with the interest “Sammlungen”, we can be less sure that she actually matches best with this TG, because this interest is less characteristic of this TG. For a user

with interest “Oper”, however, it is more likely that this is in fact the best matching group for her, because this interest is highly characteristic for this TG.

It was not our goal to implement Schulze’s model completely for performing a sociologically valid assignment of users to milieus. Instead, we wanted to show that an automatic assignment of users – even with fewer parameters and only with the milieu category “Alltagsästhetik” implemented – can be used for supporting the **collection of user interests**. As mentioned earlier, we consider the assignment of users to TGs not as our primary goal, but as a possibility to support the control of the profiling dialog.

A disadvantage of Schulze’s model is that it mainly contains those interests which help to cluster the population, i.e. which have the property of being strongly discriminating between milieus. These are often different than the interests and dislikes entered by the users in our empirical studies, however. For finally evaluating our prototype, we therefore had to enrich the TGMOD based on [Schulze 1992] by further interests, which are more conform to typical interests entered by our users (cf. Section 6.2.3.3).

Formal Representation of the TGMOD

In order to use the above described model in LINGUINI, we had to develop a formal representation for it. The most important criterion for this representation was that it had to enable the comparison of user profiles with TGs. Therefore, we decided to choose the same representation for TGs as for user profiles, but with slight modifications as described below.

As a consequence, a TG is also a **subgraph of GermaNet** extended with interest nodes. An advantage of this is that the TGMOD can be created semi-automatically by using our LA and PM components as described in Section 6.2.3.3. Like a user profile, a TG only contains GermaNet paths. No hand-crafted paths by the application designer are allowed.

Dislikes contained in Schulze’s model (e.g. “nicht Bildzeitung lesen (not reading the newspaper ‘Bildzeitung’)”) are represented in our TGMOD as interest nodes with attribute `dislike`. Schulze’s **preferences** (e.g. “klassische Musik hören (listening to classical music)”) are represented as normal interest nodes without the attribute `dislike`. The **node structure** in a TG graph is similar to the profile node structure described in Section 4.4.2.1.

The most important **differences** between TGs and user profiles are the following:

- A TG does not contain **ambiguities** – unless these are intended – whereas a profile may contain ambiguities temporarily, which can be resolved with a disambiguation strategy (cf. Section 3.6.3). In a TG, each interest/dislike is represented by exactly the GermaNet path that stands for the meaning intended by the application designer.
- In a TG, each interest/dislike contains the additional attribute `weight`, which represents the parameter **eta/Cramer’s V** in Schulze’s model, i.e. the discriminating power (see above). In our approach, this attribute is used by the DM for selecting a TG interest that has an especially high weight or relevance for generating the next question (cf. Section 5.4.2.4). The attribute `weight` does not occur in user profiles.

For illustration, a sample TG from our TGMOD can be found in Appendix A.

5.3.2 Functioning of the Target Group Matching

5.3.2.1 Target Group Matching Classes and Main Data Structures

As compared to the other components, the TGMAT has a rather simple class structure. It consists of a main class **Target Group Matching**, which implements the TGMAT algorithm and has access to the **Target Group Database**, from which it reads the TGs. The Target Group Matching class employs two further classes that represent the most important data structures: the class **Best Matching TG** and the class **Matching Result** (cf. Figure 5.12):

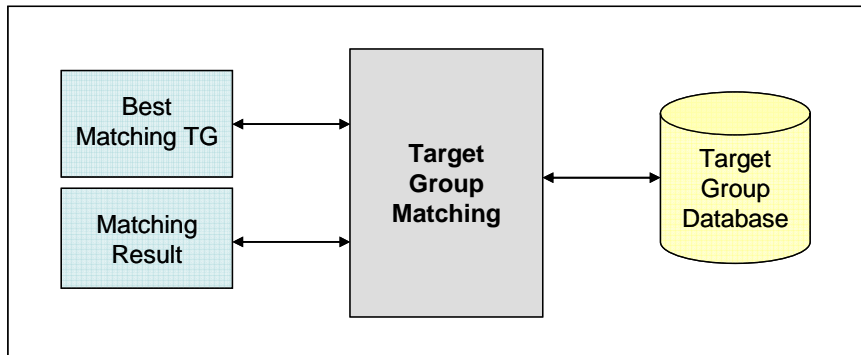


Figure 5.12: The interaction between TGMAT classes and Target Group Database

The TGMAT has the task of comparing the user profile with each TG in the TGMOD. It compares the profile with a TG by comparing each node in the profile with each node in this TG. For each such node pair, henceforth called **profile-TG-node pair**, a similarity relation (e.g. hyperonym, hyponym, siblings, no_match) between the two nodes is determined. On the basis of this similarity relation of a node pair, its numeric similarity value is calculated, which indicates the exact degree of semantic similarity between the two nodes. The overall degree of similarity between a TG and the profile is finally calculated on the basis of these similarity values.

This way, the TGMAT first computes one or several best matching TGs, i.e. TGs which are most similar to the user profile, each of which is represented by the data structure **Best Matching TG**. This data structure contains:

- A TG (i.e. its graph representation)
- All profile-TG-node pairs that have been created by comparing each profile node with each node in this TG

Each such **node pair** stored in the Best Matching TG is represented as a vector and has the following complex structure (cf. Table 5.2). The exact meaning and use of this data structure and its elements is further discussed in the succeeding sections.

Vector elements	Value	Description
1. Profile node	array with the fields 0 = object 1 = nominal_modifier 2 = activity 3 = location 4 = participants 5 = property 6 = rating 7 = frequency	A profile node is represented as an array. Each field in the array represents a specific attribute. It is instantiated with the corresponding attribute value, if set for this node. The order in the array is fix, such that field 3, for example, always represents the location attribute in all

	8 = dislike 9 = favorite_items	node pairs.
2. TG node	array with the same fields as above	The same as above, but here the fields are instantiated with values from the TG node.
3. Similarity relation	a string from the set {complete_match, hyperonym, siblings, hyponym, related_nodes, no_match}	The value describes the similarity relation between profile node and TG node in this node pair.
4. Relation value	1.0 (complete_match) 0.8 (hyponym) 0.4 (hyperonym) 0.2 (siblings) 0.025 (related_nodes) 0 (no_match)	A configurable number representing the importance of the corresponding similarity relation for the TGMAT process (cf. Section 5.3.2.4). We have indicated the values used in our prototype at the moment.
5. TG node weight	integer	The weight of the TG node (cf. Section 5.3.1.2).
6. Profile node ID	string	The value represents the GermaNet-internal node ID of the profile node, e.g. "nKognition.124.Blasmusik".

Table 5.2: Data structure representing a profile-TG-node pair

From the best matching TG(s), the TGMAT finally constructs the data structure **Matching Result** and returns it to the DM. The data structure Matching Result is a vector which contains Best Matching TG objects.

5.3.2.2 The Main Method of the Target Group Matching

The TGMAT is called by the DM with the TGMAT main method `computeBestTargetGroup()`. Figure 5.13 shows the actions performed in this method:

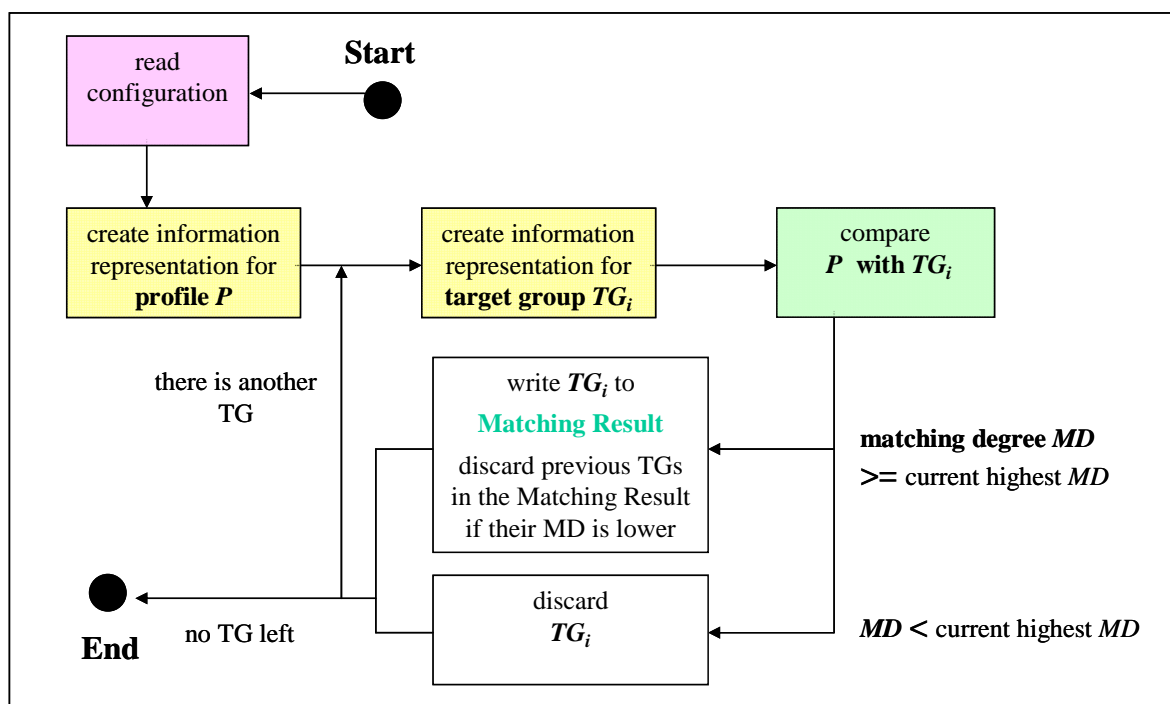


Figure 5.13: Actions of the TGMAT main method

The TGMAT uses a variety of configurable parameters (e.g. relation values, Matching Result parameter) that influence the way the TGMAT is performed. The first step is to **read this configuration** from the TGMAT configuration file (cf. top of Figure 5.13).

The next step is to extract specific **information about the profile nodes** from the user profile and to transform it into a format that is suitable for the succeeding comparison process between profile and TGs. After this, the same is done for the nodes contained in the first TG of the TGMOD. This information extraction and transformation process is described in Section 5.3.2.3.

Next, the information extracted from the profile is **compared** with that of the TG (cf. Section 5.3.2.4). A **Matching Degree (MD)** is calculated for this TG, which expresses how similar it is to the profile (cf. right hand side of Figure 5.13). If this MD is higher or equal to the currently highest MD, the TG is added to the Matching Result. Note that TGs already contained in the Matching Result that have a lower MD as the newly added TG are discarded. If the calculated MD is not higher or equal, the TG is discarded instead of being added to the Matching Result.

This information extraction and comparison process is done for all TGs in the TGMOD. When all TGs have been processed, the algorithm terminates and the Matching Result is passed back to the DM.

5.3.2.3 Extracting and Transforming Information from Profile Nodes and Target Group Nodes

In order to be able to compare the user profile with the TGs, some preparation of the information contained in the profile nodes and TG nodes has to be accomplished in advance. The main task in this preparation phase is to extract the so-called *ID-chains*, i.e. chains that consist of the latter parts of node IDs extracted from GermaNet and strung together. ID-chains are extracted from all profile and TG nodes. An ID-chain is a somewhat simplified representation of the (GermaNet) path between interest node and root. Example 5.13 illustrates the notion of ID-chains:

Example 5.13

Consider the following cutout of a profile, which contains the interest “Zeitung lesen (reading the newspaper)” and the dislike “keine Sprachen lernen (not learning languages)”:

```
<Medium id="nKommunikation.152.Medium">
  <Massenmedium id="nKommunikation.247.Massenmedium">
    <Printmedium id="nKommunikation.248.Printmedium">
      <Zeitung id="nKommunikation.249.Zeitung2"
        activity="lesen"
        object="Zeitung"/>
    </Printmedium>
  </Massenmedium>
  <Sprache id="nKommunikation.1104.Sprache"
    activity="lernen"
    dislike="keine"
    object="Sprachen"/>
</Medium>
```

The ID-chain extracted for the interest is shown in (a) and for the dislike in (b):

- (a) 152.Medium.247.Massenmedium.248.Printmedium.249.Zeitung2.
- (b) -152.Medium.1104.Sprache.

They have been created from the parts of the profile shown in bold type, i.e. the last part of each node ID in the path up to the topmost node “Medium”.

As can be seen in Example 5.13 (b), dislikes are distinguished from interests by means of a **minus sign**, which is added at the beginning of the ID-chain. This distinction is important in order to avoid the situation that an interest in the profile is mistaken to be similar to a dislike in the TG or vice versa.

Note that this preparation has to be done only once for the TGs (because – as opposed to user profiles – they do not change during the course of a dialog). For the profile, however, this preparation has to be done anew after each user answer in the dialog in order to take into account changes in the profile.

This is not the only information extracted in the preparation phase, however. The following information is also extracted from profile nodes and TG nodes:

- The attributes of the node
- The GermaNet ID of the profile node (e.g. “nKognition.120.Rock”)
- The GermaNet IDs of its subnodes
- The node weight (which is only set in the TGs, not in the profile)

We will see in the next section for which purposes this information is used.

5.3.2.4 Determining the Most Similar Target Group

The central idea of the TGMAT is that the similarity between a profile and a TG is calculated on the basis of its node similarities. For this purpose, each node in the profile is compared with each node in the TG. In other words, we consider pairs consisting of a profile node and a TG node. For each such node pair, we calculate a *similarity value* which expresses the degree of similarity between the nodes in this pair. The overall degree of similarity between profile and TG is called the *Matching Degree* and is calculated on the basis of the similarity values of all their node pairs.

Similarity Relations and Relation Values

The *similarity relation* between two nodes is calculated by means of their ID-chains, described in the previous section. Table 5.3 below (column 3) shows all similarity relations that can occur in our approach, listed with descending similarity.¹⁰ The two columns on the left give an example for a specific node pair (e.g. “Segeln” and “Tauchen” in row 4). The similarity relation of this node pair is indicated in the third column (e.g. “Segeln” is a sibling of “Tauchen”, because both have the same parent node in the graph). The rightmost column shows the configurable *relation value* (not to be confused with the *similarity value* described in the next section) as it is set at the moment in our prototype system. For better readability, the ID-chains in the first two columns have been shortened to numbers (e.g. “1004.2422.2428”):

¹⁰ For a description of hyperonym, hyponym, and other GermaNet relations cf. Section 3.5.

Example: ID-chain and TG node name	Example: ID-chain and profile node name	Similarity relation	Relation value
1004.2422.2428 Segeln (sailing)	1004.2422.2428 Segeln (sailing)	complete match of the entire ID-chains (including the interest node names)	1.0
1004.2422.2428 Segeln (sailing)	1004.2422 Wassersport (water sports)	profile node is hyperonym of TG node	0.4
1004.2422 Wassersport (water sports)	1004.2422.2428 Segeln (sailing)	profile node is hyponym of TG node	0.8
1004.2422.2428 Segeln (sailing)	1004.2422.2433 Tauchen (diving)	profile node is sibling of TG node	0.2
720.517.433 Sachbuch (specialized book)	720.517.218.33 Autobiographie (autobiography)	profile node is a related node of the TG node, i.e. it is in the same subtree that has a depth of 2 nodes at most	0.025
1004.2422.2428 Segeln (sailing)	334.324.15 Zeitung (newspaper)	no match at all between the two paths	0

Table 5.3: Similarity relations and relation values

It is important to note that this way we are able to consider various types of semantic relations between nodes in the profile and TG graphs, not only the cases `complete_match` and `no_match`. This approach enables us to take into account the graph structure for finding a suitable node pair for asking the next question about. This is particularly important for the system’s capability of introducing new topics that are related to known user interests during the dialog.

In the following, we explain why we assume the **relation values** shown in the rightmost column in Table 5.3 for these similarity relations. All relation values lie in a range between 1 and 0 and can hence also be interpreted as percentages indicating the degree of similarity. The cases **complete_match (1.0)** and **no_match (0.0)** mark the boundaries of this range and are self-explanatory. In the **hyperonym** case, we assign only **0.4**, since the interest in the profile (e.g. “Wassersport (water sports)”) is more general and does not necessarily imply the more specific interest in the TG (e.g. “Segeln (sailing)”). In the **hyponym** case, we assign **0.8**, since an interest in sailing implies an interest in water sports to some degree, but maybe not in all kinds of water sports (therefore we did not assign value 1.0 here). In the **siblings** case, we only assign the value of **0.2**, since siblings can have quite different meanings (e.g. “Techno” and “klassische Musik (classical music)”). We cannot assume that if a user likes Techno music, she will probably also like classical music. Therefore, we assign a rather low value here, which still reflects some relatedness of the common topic (e.g. music), however. **Related nodes** are nodes that occur in the same subtree which has a depth of 2 nodes at most. The interests may occur on arbitrary levels in this subtree. Related nodes have a similarity degree (**0.025**) that is even weaker than that of siblings. We have included them with a higher value than 0, however, because the probability is high that the nodes in this same subtree are

semantically more related than nodes that are very far from each other in the graph. These currently used configuration settings for relation values have been determined empirically by investigating our input data set. They can be changed by the application designer, if desired.

Determining the Similarity Relation and Similarity Value of a Node Pair

The **similarity relation** of a node pair is determined by comparing each sign in the ID-chains of profile node and TG node. The concordant signs are counted from left to right. The counting stops if a non-concordant sign is reached. Example 5.14 illustrates this:

Example 5.14

Given to ID-chains

```
113.Musikstil.130.Pop.    (profile node)
113.Musikstil.120.Rock.  (TG node)
```

The number of concordant signs is 15 here (in bold type below). The counting stops at position 15 in the ID-chains (indicated by the bar):

```
113.Musikstil.1 | 30.Pop.
113.Musikstil.1 | 20.Rock.
```

Simultaneously, the number of “concordant dots” is counted. In Example 5.14, the two ID-chains have two “concordant dots”, contained in the concordant part “113.Musikstil.1”.

In order to determine the similarity relation (e.g. `siblings` in Example 5.14), the **number of concordant signs** c is compared with the length of the two TG ID-chains. If both ID-chains have the same length, for example, and c is equal to this length, then the similarity relation is a `complete_match`. If the two paths are concordant except for the last node, for instance, then their similarity relation is `siblings`. If c is equal to the length of the profile node ID-chain, for example, but smaller as the TG ID-chain length, then the profile node is a `hyperonym` of the TG node, and so on. We deviate slightly from the strict notions of `hyperonym` and `hyponym` in GermaNet here, which only refer to direct child nodes or parent nodes. By only checking for a greater or smaller c , we do not only get direct child nodes or parent nodes, but also nodes that are further up or down in the path.

In the next step, the **similarity value** sv , a numeric value indicating the exact degree of similarity between the two nodes, can be calculated with the following formula:¹¹

$$sv = \frac{rel_val \cdot num_conc_dots}{num_TGNode_dots}$$

In this formula, rel_val is the relation value of the similarity relation of the two ID-chains (taken from the configuration settings, cf. Table 5.3 above). Num_conc_dots is the number of concordant dots that occur in both ID-chains, which has been counted before. Num_TGNode_dots is the number of dots in the TG ID-chain. The division with num_TGNode_dots is made in order to calculate the proportion between the two ID-chains. In Example 5.14, the similarity relation is `siblings` and hence rel_val is 0.2 (cf. Table 5.3 above). The similarity value sv for the two ID-chains in Example 5.14 is $sv = 0.2 * 2 / 4 = 0.1$.

¹¹ Note that this formula can be used for all similarity relations, except for `hyponym`. For `hyponym`, the denominator has to be $num_ProfileNode_dots$ (i.e. the number of dots in the profile node ID-chain).

It is important to note that we not only compare the specific node names with each other, but the entire GermaNet **synonym set** a node name is an element of. This way, we are able to find matches for two words with the same meaning, even if their word forms are different. The number of matches between profile nodes and TG nodes is hence higher, than if only the identity between word forms were checked.

Special Case: Dislikes

In Section 5.3.1.1, we introduced the **dislike parameter**, which is set to “true” for TGMODs and profiles that not only contain interests, but also dislikes. In the following, we explain what happens in the TGMAT, if this is the case. The comparison of an **interest node** and a **dislike node** has to be handled differently than the comparison of two interest nodes.

Remember that the ID-chain of a dislike has a minus sign added (cf. Section 5.3.2.3). In case the algorithm detects an ID-chain pair in which only one ID-chain has a minus sign, it checks whether one of the two ID-chains is the **exact complement** of the other (i.e. the entire paths are identical except for the minus sign) as shown in Example 5.15:

Example 5.15

```
113.Musikstil.130.Pop. (profile node)
-113.Musikstil.130.Pop. (TG node)
```

In this case, the similarity value sv is set to **-1.0** and the comparison for this TG terminates immediately. In all other cases, the similarity value is set to **0** (i.e. we do not count these pairs for the final Matching Degree of the TG) and the algorithm continues with the comparison of the next ID-chain pair for this TG. The immediate termination of the comparison process for a TG in which an exact complement has been found is due to the following reason. In our opinion, a TG should not become the most similar TG of a user profile, if it contains one or several dislikes that are complementary to user interests (or vice versa) (as in Example 5.15). In this case, the group profile seems to be semantically contradictory to the user profile in some respects. Hence, this TG disqualifies itself immediately, if the first contradiction of this type is detected.

Calculating the Matching Degree between a Profile and a TG

We will now explain how the similarity values determined for the node pairs so far add up to the final Matching Degree. The **Matching Degree (MD)** of a TG indicates how similar this TG is to the profile. An MD is calculated for each TG. Finally, the TG with the highest MD constitutes the **Matching Result**. It is possible that several TGs have the same (highest) MD. Then all of these TGs together constitute the Matching Result. The Matching Result is the final result of the TGMAT, which is returned to the DM.

The MD of a TG is determined as follows. Each of the m profile nodes has been compared with n TG nodes (i.e. their ID-chains have been compared). The result of this comparison is – for each profile node – a list of n node pairs. Each of the n node pairs has a similarity value sv . For each profile node, the **maximum similarity value sv_{max}** of this list of n node pairs is selected.

Figure 5.14 illustrates this by means of an example, which assumes a simple TG with only 11 TG nodes. The profile node “Musik” (i.e. its ID-chain) has been compared with each of the $n = 11$ nodes in the TG (i.e. with their ID-chains), which results in $n = 11$ node pairs for “Musik”. The node pair “profile: Musik – TG: Musik” is shown in bold type in Figure 5.14. This node pair has a `complete_match` similarity relation (with relation value 1.0). Its similarity value sv can be calculated by means of the formula we have presented above. Because of its high relation value (1.0), this node pair has the highest similarity value of all 11

node pairs shown in Figure 5.14 and is hence selected as *sv_max* for the profile node “Musik”. The *siblings* node pair “Musik – Tanz”, for example, has a lower *sv* value and hence cannot be selected as *sv_max*. The whole procedure is repeated for every profile node.

profile-TG-node pair		similarity relation
Profile: Musik,	TargetGroup: Tennis,	nomatch, 0.0
Profile: Musik,	TargetGroup: Skifahren,	nomatch, 0.0
Profile: Musik,	TargetGroup: Surfen,	nomatch, 0.0
Profile: Musik,	TargetGroup: Tanz,	siblings, 0.2
Profile: Musik,	TargetGroup: Sachbuch,	nomatch, 0.0
Profile: Musik,	TargetGroup: Modezeitschrift,	nomatch, 0.0
Profile: Musik,	TargetGroup: Cafe,	nomatch, 0.0
Profile: Musik,	TargetGroup: Musik,	completematch, 1.0
Profile: Musik,	TargetGroup: Film,	nomatch, 0.0
Profile: Musik,	TargetGroup: Fernsehen,	nomatch, 0.0
Profile: Musik,	TargetGroup: Zeitung,	nomatch, 0.0

Figure 5.14: Selecting the node pair with *sv_max* from all node pairs built for profile node “Musik”

Finally, we have selected the *sv_max* value *m* times, i.e. one for each of the *m* profile nodes. The final Matching Degree *md* can now be calculated by means of the following formula:

$$md = \frac{\sum_{i=1}^m sv_max_i}{n}$$

In this formula, the *m* values for *sv_max* are added up. This sum is divided by the number *n* of TG nodes to normalize the result. This normalization is necessary to compensate for the different sizes of the TGs (i.e. the different number of nodes contained in them).

The final **Matching Result** is created as follows. The *md* calculated for the first TG is compared with that calculated for the next TG. Two cases have to be distinguished:

- If one of the two TGs has an *md* that is 0.0125 higher than the *md* of the other TG, the TG with the higher *md* is added to the Matching Result and the other TG is discarded.
- If the difference of the two *md* values is smaller than 0.0125, both are considered most similar to the profile and are added to the Matching Result.

The *md* of each further TG is compared with the **currently highest *md*** and added to the Matching Result or discarded according to this schema.

We have conducted many tests for determining the practically best value for the (configurable) Matching Result parameter, which proved to be 0.0125. If the value is too low (e.g. 0), fewer TGs are added to the Matching Result than are actually similar to the user profile, which may lead to a smaller set of possible next questions in the dialog (cf. Section 5.4). This may cause problems for the DM in finding a suitable next question. If the value is set too

high, too many TGs are added to the Matching Result, which may finally lead to the point where all TGs match the profile equally well and the entire TGMAT process becomes redundant.

5.4 The Dialog Manager

After having introduced the functions of the TGMAT, we can now address the central system component, the **Dialog Manager (DM)**. The DM constitutes the core of the *dialog management* part of the system. This section is structured as follows. Section 5.4.1 first gives an overview of the DM. We briefly present its tasks and embedding into the profiling system and explain the interaction between the DM and the **User Interface component (UI)**.

Section 5.4.2 discusses the functioning of the DM component in detail. First, the DM classes and main data structures are introduced. We then describe the DM main method and the tasks that are accomplished for each dialog unit. In the remaining sections, two more complex tasks of the DM are explained in full detail: the processing of user answers by triggering adequate LA and PM actions and the selection of suitable next questions.

5.4.1 Overview of the Dialog Manager

5.4.1.1 Tasks and Embedding of the Dialog Manager

As the main system component, the DM is responsible for controlling both the dialog interaction with the user and the construction and update of the user profile. The DM distinguishes two basic situations, which we have introduced in Section 4.2.2:

1. **Constructing an initial profile** from the starting answer of the user:

The DM triggers the LA to analyze the answer given by the user to the starting question. Then, it passes the user interests the LA has extracted from this answer to the PM, which uses them to build a profile graph.

2. **Refining dialog** with the user and profile update:

The DM has to select suitable questions for acquiring further profile information. These questions have to be adaptive to the interests mentioned by the users. The answers received have to be analyzed and the profile has to be updated accordingly.

For **constructing an initial profile**, the DM mainly delegates tasks to the LA and PM. The real tasks of the DM are related to the **refining dialog**. In the refining dialog, the interaction with the user consists of dialog units. A *dialog unit* (cf. Section 4.2.2.2), in our approach, comprises the processing steps between the last user answer and the next question. In these processing steps, the DM interacts with the other components to accomplish the following main tasks:

- Analyze the user answer and extract profile information
- Construct or update the profile with the information
- Select the contents for the next question
- Make sure that a question with these contents has not been asked before
- Generate the final natural language question

Figure 5.15 illustrates – in a simplified form – with which components the DM interacts and which information packages are passed between them (the main system components are marked in grey):

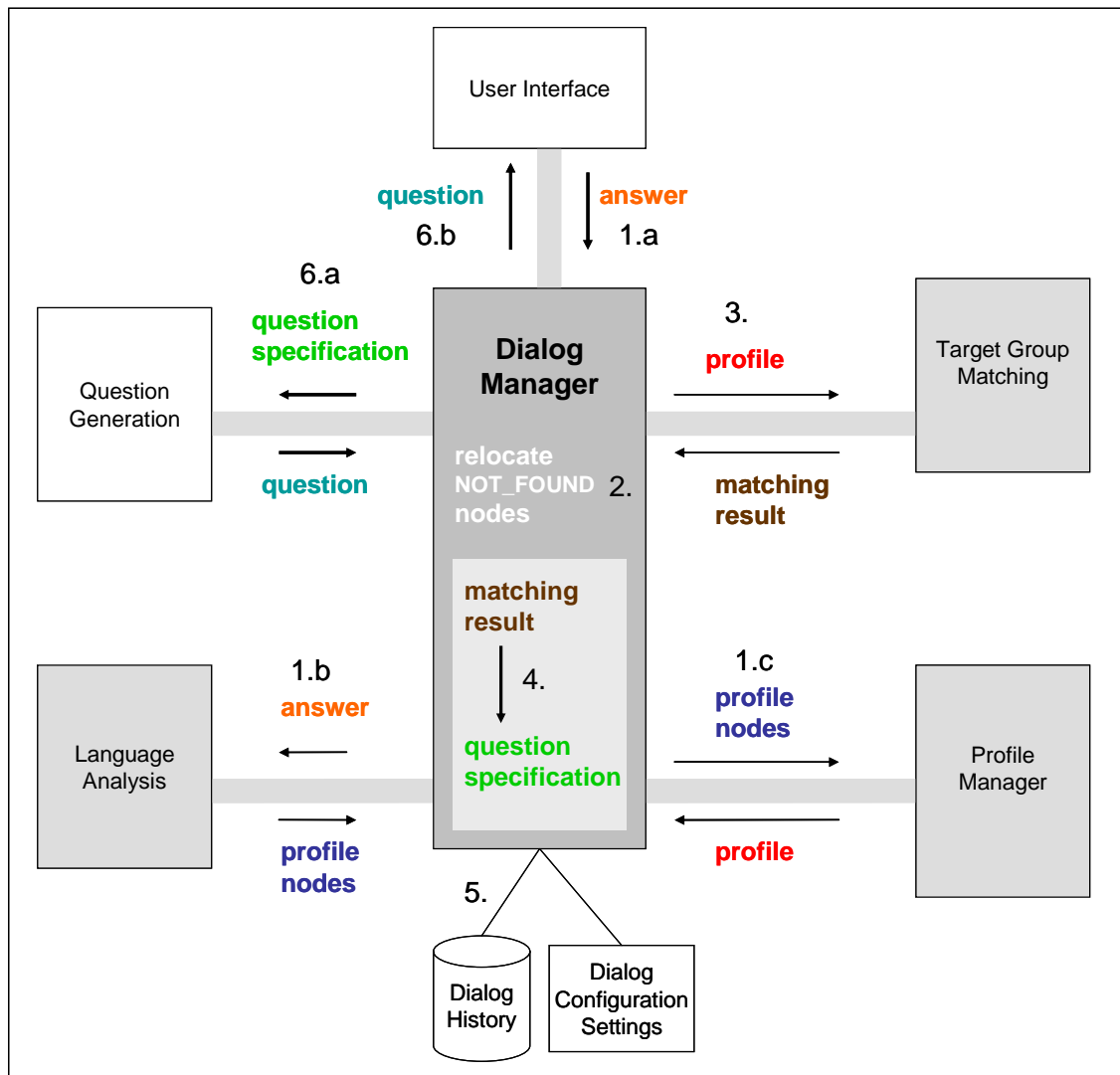


Figure 5.15: DM tasks and information packages passed between DM and other components

The main tasks mentioned on the previous page consist of a number of subtasks related to specific system components, which are indicated by means of their numbers in Figure 5.15:

1. a) An **answer** is received from the UI. b) The answer is analyzed by the LA which returns a set of profile nodes. c) The profile nodes are passed to the PM for constructing or updating a profile.
2. By means of the TGMOD, the DM tries to **relocate nodes** in the profile it could previously not find in GermaNet (“NOT_FOUND nodes”). The aim is to insert them in the semantically right position in the profile graph.
3. The DM triggers the **TGMAT** for the current state of the profile and receives back the Matching Result.
4. From the Matching Result, the contents for a **suitable next question** are selected and are represented by a formal Question Specification.

5. The DM checks the **Dialog History** to make sure that a question with these contents has not been asked before. In case such a question has been asked before, the process continues with the second-best suitable question and so on. If an unused Question Specification is found, it is stored in the Dialog History.
6. a) The **Question Generation** creates the final natural language question from the Question Specification. b) The DM passes the question to the UI.

The DM repeats these six tasks for each dialog unit until the dialog end is reached. Figure 5.15 also shows the Dialog Configuration Settings, which are read at the beginning of the dialog process.

5.4.1.2 The Interaction between Dialog Manager and User Interface

For communicating with the user, the DM interacts with the **User Interface (UI)**. The main task of the UI is to present natural language questions to the user and to receive the user's answers. Previously, we explained that the UI of LINGUINI can either be realized as a Web interface which can be used via keyboard and display or as a speech interface which can be used via headset/speakers and microphone.

For evaluating our prototype system, we chose the Web interface variant with typed input and output. We have implemented the UI as a JAVA Servlet, which **initializes the DM** when the user logs into the system by means of an Internet browser. During its initialization, the DM initializes in turn all other system components and reads the configuration settings and the TGs that belong to the TGMOD. After the DM has been initialized, the steps performed by the UI are the following:

1. Get **starting question** (cf. Section 4.2.1.1) from DM and display it for the user.
2. Read user answer.
3. Pass user answer to DM and get next question from DM.
4. Display the next question for the user.

Steps 2 and 4 loop until the DM sends the **closing remark** (cf. Section 4.2.2.2) back to the UI, instead of a next question. The closing remark is displayed for the user and the dialog ends. The UI stores the entire dialog course in a log file, which the user can see during and after her interaction.

The UI we have used for evaluating the prototype is shown in Section 6.3.1.3. It has a simple design, because we intended to evaluate the functionality of the dialog rather than the usability of the UI. For the use in E-Commerce platforms, this UI can be replaced by various types of graphical user interfaces, e.g. an animated virtual character or agent that talks to the user (cf. Sections 2.1.1.2 and 7.2.1). The choice of interface strongly depends on the application domain and should not be dictated by the profiling system. We therefore separated the main functionality of LINGUINI clearly from the test UI, such that an easy exchangeability of the UI is guaranteed.

5.4.2 Functioning of the Dialog Manager

5.4.2.1 Dialog Manager Classes and Main Data Structures

Figure 5.16 shows the DM classes. The class **Dialog Manager** is the central class, which also contains the DM main method. It is assisted by the **Question Selection** class, which is responsible for selecting the contents for a suitable next question from the Matching Result. In

case it is necessary to compare the attribute values in a profile node with those of a TG node, the Question Selection class calls the Attribute Comparison class. The **Attribute Comparison** class contains a set of conditional rules that are used to create different Question Specifications on the basis of specific attribute value settings. Both Question Selection and Attribute Comparison have access to the **Question Frames** for constructing Question Specifications. The DM class maintains a **Dialog History**, in which it stores information about questions that have been asked and which it uses to make sure that no identical question is asked twice.

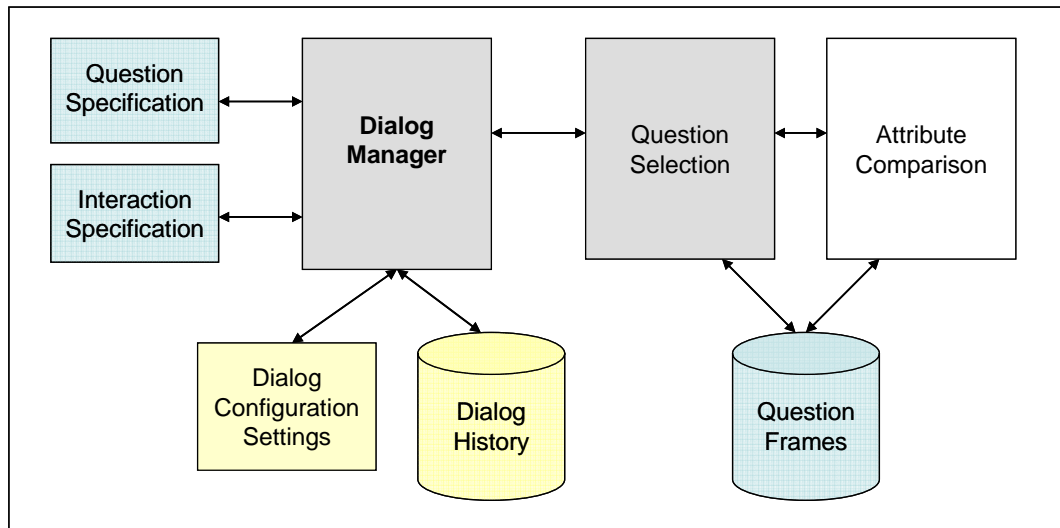


Figure 5.16: Interaction between DM classes, DM resources, and configuration settings

As described in Section 4.2.2.2, various dialog parameters can be configured. The DM class is also responsible for reading the **Dialog Configuration Settings** at the beginning of the interaction process. The following main parameters that influence the course of the dialog can be configured by the application designer (“black box parameters”) and have been described in Section 4.2.2.2 in detail:

- Dialog length (i.e. maximum number of questions asked)
- Nodes with a high application priority
- Attributes with a high application priority

There is a variety of “glass box parameters”, which also strongly influence the dialog course and should only be configured by experienced LINGUINI developers. They are described in the remaining sections in connection with the tasks for which they are used.

Figure 5.16 above also shows the main data structure of the DM, the **Question Specification (QSpec)**, represented by the Question Specification class. Figure 5.17 below illustrates this data structure in more detail by means of a simple example. More complex QSpecs are shown in Examples 5.21 and 5.23 in Section 5.4.2.4. Here, we only briefly mention for which tasks the information in a QSpec is needed. They will be described in detail later on.

<p>Question frame: „A propos pObject: Mögen Sie tgFavoriteItems?“</p> <p>Profile node: [pObject=Bücher, pActivity=lesen, ...]</p> <p>Target group node: [tgObject=Bücher, tgActivity=lesen, tgFavoriteItems=Krimis, ...]</p> <p>Instructions for answer processing: if answer = „ja“ then instruction = add_TGNode[9] (i.e. add “Krimis” as value of favorite_items attribute to this profile node) if answer = „nein“ then instruction = add_TGNode[9]_dislike (i.e. add “Krimis” as value of dislike attribute to this profile node)</p> <p>Profile node ID: nArtefakt.2711.Buch</p> <p>Answer type: yes/no</p> <p>Preconditions for this QSpec: <empty></p>

Figure 5.17: Sample Question Specification

A QSpec contains all information related to the question that is supposed to be generated. First of all, this is the **question frame**. Figure 5.17 (and also Examples 5.21 and 5.23 in Section 5.4.2.4) show possible question frames. Question frames contain variable slots that need to be filled with information about the node pair (“pObject” and “tgFavoriteItems” in Figure 5.17). Therefore, the QSpec also contains the **profile node** and **TG node** contained in the node pair, more precisely their attribute values. These attribute values are used by the Question Generation later on to generate the final natural language question. For this purpose, the Question Generation inserts the attribute values into the question frame slots. The QSpec also contains **instructions for answer processing**. The **profile node ID** is needed in the QSpec for locating the node in the profile. This is necessary in order to execute the instructions for answer processing. Finally, the QSpec contains the expected **answer type** of the question (cf. Section 4.2.1.2) and also a **precondition** which indicates the circumstances under which this QSpec may be applied. The function of the individual QSpec elements will be described in more detail in the remainder of this chapter.

The data structure **Interaction Specification (ISpec)** stores QSpecs that can be used for the next interaction step with the user. The idea behind the ISpec is to bundle all information needed for the next dialog unit in a single data structure. The Dialog History, for example, consists of ISpecs that have been used in the dialog. In our approach, **at most two QSpecs** are stored in an ISpec. Typically, an ISpec contains exactly one QSpec, but sometimes two questions belong together more closely in the discourse and their QSpecs are therefore stored together in a common ISpec. In this case, the second question in the ISpec may only be asked if the user answer to the first question fulfills certain conditions (e.g. if it has been negated by the user). An example for an ISpec which contains two QSpecs is given in the next section (cf. Section 5.4.2.2 “Task 6”).

5.4.2.2 The Main Method of the Dialog Manager

The main method of the DM is `getNextQuestion()`, which is called by the UI as described in Section 5.4.1.2. In this method, the DM receives from the UI the last natural language user answer and it returns to the UI the next natural language question. This method controls and triggers the six tasks we presented in Figure 5.15 in Section 5.4.1.1.

This method furthermore controls the **termination of the dialog**. The DM counts the questions it has already asked the user. Before each execution of its processing steps, the main method checks whether the maximum number of questions has been reached. It also checks whether the user has entered “ende (end)” instead of an answer, which is the command for terminating the dialog. Finally, the dialog also ends, when a certain number of fallback questions have been asked. If one of the three cases applies, the DM sends the closing remark (cf. Section 4.2.2.2) to the UI and the dialog ends.

If there is no reason to end the dialog, the main method checks whether the current **answer refers to the starting question**, i.e. whether it is the first answer it has received. In this case it has to analyze the starting answer and **construct an initial profile** for this user. More precisely, it has to perform the following tasks (they are described in detail below):

1. Call LA for answer analysis and PM for construction of initial profile.
2. Relocate NOT FOUND nodes.
3. Trigger TGMAT.
4. Select most suitable next question and create queue of further possible questions.
5. Check Dialog History.
6. Trigger Question Generation.

The result of the main method is the final natural language question, which is passed back to the UI. After the UI has displayed the question and received the user’s answer, it calls the DM main method again for this answer. The DM “knows” that the first answer has been processed already and, in consequence, the **refining dialog** starts now. It consists of the following tasks:

1. Execute the instructions contained in the QSpec of the last question.
2. Relocate NOT FOUND nodes.
3. Trigger TGMAT.
4. Select most suitable next question and create queue of further possible questions.
5. Check Dialog History.
6. Trigger Question Generation.

Again, the result (i.e. the question) is returned to the UI which proceeds as described in Section 5.4.1.2. These six tasks are performed for each dialog unit until the **end of the dialog** is reached. The disambiguation described in Section 3.6.3 is not yet integrated here, but could be inserted between Tasks 2 and 3, for example, such that the TGMAT and question selection would be performed for an already disambiguated profile.

The only difference between the initial profile construction and the refining dialog lies in the first of the six tasks. We will explain all tasks in detail now. Task 1 and Task 4, however, are so complex that they will only be summarized here and explained in full detail in Sections 5.4.2.3 and 5.4.2.4.

1. Answer Processing by means of Language Analysis and Profile Manager

In case a **new profile needs to be constructed**, the DM simply calls the LA to process the user answer and the PM to construct the profile from the information extracted from the user answer. During the **refining dialog**, the DM follows the instructions contained in the QSpec of the last question (for which it has just received an answer). The instructions check whether the user answer meets certain conditions and indicate how to process the answer and store the information in turn. The instructions are directed towards LA and/or PM. They are described in detail in Section 5.4.2.3.

2. Relocate NOT FOUND Nodes

Whenever the system cannot find a user interest in GermaNet, the interest is stored under a special node called “NOT FOUND” in the profile graph (cf. Sections 5.2.2.2 and 5.2.2.3). During the refining dialog, the system regularly tries to **relocate NOT FOUND nodes** in the profile by exploiting related information stored in the TGMOD. For this purpose, the `object` value of the NOT FOUND node is compared with the `favorite_items` value of all interests in the TGs. The aim is to find a `favorite_items` value that is identical to this `object` value. If such a TG is found, it is checked as to whether this node possesses an `object` or `activity` value. If this is the case, the NOT FOUND node in the profile is enriched with this `object` or `activity` information and can then be searched for in GermaNet again. Since TGs only contain nodes that have been found in GermaNet, this search will be successful and all paths found in GermaNet can be inserted together with the node at the semantically right position in the graph.

3. Trigger Target Group Matching

In the next step, the TGMAT is triggered for the current state of the profile and is accomplished as described in Section 5.3. The Matching Result, which contains all profile-TG-node-pairs of the profile and the most similar TG(s), is returned to the DM. The Matching Result serves as input for the next step.

4. Select most Suitable Question and Create Question Queue

The next task of the DM is to select from the Matching Result the most suitable profile-TG-node pair for asking the next question about. It creates one or two QSpecs for this node pair, which it stores in an ISpec (cf. Section 5.4.2.1). The ISpec is stored as the first element in a question queue. Then, the DM selects a number of further suitable profile-TG-node pairs from the Matching Result, also creates ISpecs for them, and adds them to the question queue. In the queue the ISpecs are ordered according to their suitability for the next question at this stage of the dialog. The idea behind using a question queue is that the DM can fall back to other suitable ISpecs in case the mostly preferred ISpec has already been used in the dialog (cf. “5. Check Dialog History” below). This entire Task 4 is accomplished by the Question Selection class and is described in detail in Section 5.4.2.4.

5. Check Dialog History

The **Dialog History** is a database in which – for each user – all ISpecs that have been used in the dialog with this user are stored. The ISpecs are stored for each user separately in the database and are distinguished by means of the Profile ID. Whenever a new ISpec is going to be used in the dialog, it is added to the Dialog History. The main purpose of the Dialog History is to make sure that no question is asked twice in the dialog.

The Dialog History check is performed by the method `checkHistory()`. This method takes as input a vector of ISpecs, which has been created in the previous step (cf. “Task 4” above). This vector contains a specific number of ISpecs, which represent further possible questions. The number of elements in this vector is configurable and has currently been set to ten at most in our prototype implementation. In this vector, the ISpecs are already ordered according to their priority, i.e. the most suitable question is element one, the second-best suitable question is element two, and so on. The method `checkHistory()` checks, whether the first ISpec in the vector (i.e. the most suitable question) is already contained in the Dialog History. If this is the case, it checks whether the second-best suitable question is contained, and so on. It returns the first ISpec which is not yet contained in the Dialog History (the rest of the ISpec vector remains unchecked). This ISpec will be used to generate the next question for the user. The returned ISpec is also stored in the ISpec after the Dialog History check.

6. Trigger Question Generation

After the DM has verified that the current ISpec has not been used in the dialog before by means of the Dialog History check, the QSpec information contained in the ISpec can be passed on to the Question Generation component. The Question Generation has the task of generating the final natural language question for the user from a QSpec (cf. Section 3.8.2). For this purpose it uses from the QSpec information the question frame and its variable settings that depend on the profile-TG-node pair about which the question will be asked.

Typically, an ISpec contains only one QSpec. In this case, this QSpec is simply passed to the Question Generation. In Section 5.4.2.1, however, we have mentioned that in some cases an ISpec may contain two QSpecs. This special case is more difficult for the DM, because it has to execute the six tasks in a different manner. If an ISpec contains two QSpecs, the DM has to remember that it has used the first QSpec for generating the preceding question and it has to check the user’s answer to this question. If the answer meets certain preconditions, the second QSpec will be asked right after the first one.

The six tasks are executed as follows in this case: The first QSpec in the ISpec has been used for asking a question and the user answer has been received. The DM remembers that this is in fact an ISpec which contains two QSpecs. The user answer is processed (Task 1) and NOT FOUND nodes are relocated (Task 2). Now, the DM checks whether the user answer meets the preconditions (see below) for also asking the second QSpec. If this is the case, Tasks 3-5 are omitted and the QSpec is passed directly to the Question Generation (Task 6). If this is not the case, the second QSpec is discarded and Tasks 3-6 are performed as usual, i.e. a new set of suitable next questions is selected. Since always the entire ISpec is checked against the Dialog History before the first of its QSpecs is used, it is guaranteed that also the second QSpec has not been used in the dialog before.

For realizing this functionality, we have integrated a **preconditions parameter** into the QSpec. This parameter defines which preconditions have to be fulfilled in order for the QSpec to be usable at a given state of the dialog. For all QSpecs that occur alone in an ISpec, this parameter is empty, i.e. no particular preconditions have to be met for asking the question. If two QSpecs occur together in an ISpec, the preconditions parameter is set for the second QSpec. A precondition may be, for example, that the user has answered “nein (no)” to the last question (cf. Example 5.16). If the user has answered “ja (yes)”, the second QSpec may not be used for the next question as a consequence. Example 5.16 illustrates that two QSpecs that occur together in an ISpec are also closely connected with respect to their contents:

Example 5.16

An ISpec contains the following two QSpecs. The use of QSpec2 depends on the answer to QSpec1.

QSpec1: precondition is empty

Frame: “Zum Thema pObject: Interessieren Sie speziell tgProperty pObject?”

Example: “Zum Thema Vasen: Interessieren Sie speziell chinesische Vasen? ”

(“With respect to vases: Are you especially interested in Chinese vases?”)

QSpec2: precondition: <user answer = “nein (no)”>

Frame: “Für welche Art von pObject interessieren Sie sich dann?”

Example: “Für welche Art von Vasen interessieren Sie sich dann?”

(“So, what kind of vases do you find interesting?”)

Which QSpecs may occur together and which only occur alone is defined in the Question Selection Schema and the Attribute Comparison Rules (cf. Section 5.4.2.4). The idea behind having ISpecs with two QSpecs is that in some dialog situations it is easy to acquire some additional information about a specific topic by asking a very closely related question next. For instance, we might want to ask a *second* question about the attribute `property` as in Example 5.16. Normally, the system only asks *one* question per attribute. If we want to ask a second question about an attribute, we hence have to use two QSpecs in a common ISpec. But also in other situations, in which we want to ask another question that is very closely related to the topic of the previous question, it is easier to use a second QSpec instead of performing the TGMAT and question selection process again. Furthermore, with having the system select the next question (by means of TGMAT and question selection) we cannot guarantee that this next question sticks very closely to the given topic, since the system may choose another topic for continuing the dialog.

5.4.2.3 Instructions for Answer Processing

In the previous section, we mentioned that Task 1 and Task 4 of the DM require a separate discussion due to their complexity. In this section, we explain Task 1 in more detail, which is concerned with the processing of the user answer. This process is controlled by the DM, which delegates specific subtasks to LA and PM.

The process of analyzing the starting answer and initially constructing a profile for a new user has been explained in the previous section already (cf. Section 5.4.2.2 “Task 1”) as well as in Sections 5.1 (LA) and 5.2 (PM). We therefore concentrate on Task 1 during the **refining dialog** here. During the refining dialog, the *instructions for answer processing* (short: *instructions*) contained in a QSpec play a major role. They help the DM to decide which kinds of LA and PM actions to trigger in order to make use of the information contained in the user answer.

Assume that the last question asked was q_i (created from $QSpec_i$) and that the DM has received an answer a_i to this question. Now, the DM first reads the instruction(s) in $QSpec_i$. The instructions indicate whether LA and PM together or only the PM will be involved in the processing of a_i . They also specify which operations the PM has to perform on the profile in order to insert the newly acquired information.

The Number of Instructions Contained in a QSpec

According to the **answer type** (cf. Section 4.2.1.2) set in the QSpec, two cases have to be distinguished:

- For type “**ja/nein (yes/no) answer**”, the QSpec always contains **two instructions**: the first one for the case that the answer is “ja (yes)”, the second one for “nein (no)”.
- For the types “**free user answer**” and “**selected value answer**”, the QSpec contains only **one instruction**.

For a “ja/nein (yes/no) answer” we need two instructions for the following reason. If the user has answered “yes” in a_i , this means that usually some information is added to an interest or a new interest is created. If the user has answered “no” in a_i , this usually means that she dislikes something. In this case the `dislike` attribute has to be set in an interest node or a new node which is a dislike has to be created. In order to be able to handle interests and dislikes differently, we distinguish between agreeing and disagreeing answers here and react to them with different instructions.

Executing Instructions

The DM now executes the instruction(s), which may involve one or both of the following steps:

1. **Only for instructions that occur in a QSpec with answer type “free user answer”:**
Call the LA to extract new profile information from answer a_i .
2. **For instructions that occur in a QSpec with other answer types:**
Call the PM for inserting the newly acquired information into the profile in the specific way indicated by the instruction.

Note that Step 1 (calling the LA) is not contained in all instructions. Whether it is contained depends again on the answer type set in the QSpec. If the type “**free user answer**” is set in the QSpec, we process the user answer with the LA in order to capture all information contained in it. This is important, because the user might reveal more information voluntarily than just the attributes we have asked for and we do not want to lose this additional information. In the case of the types “**ja/nein (yes/no) answer**” or “**selected value answer**”, we do not need to call the LA, because it is obvious how the information given in the user answer has to be related to specific attributes. In this case, however, another way to compose an interest node has to be found (remember that his task is normally also accomplished by the LA). For this reason, the DM is able to compose the node itself in this situation. It then passes the node to the PM, which inserts it into the profile.

Instructions are identified by means of their names (e.g. “`add_TGNode[9]`” in Figure 5.17, Section 5.4.2.1 above). These names are set in the QSpec. The DM possesses a set of **instruction processing rules** that have the instruction name in the conditional part and the actions to perform in the body (or: consequential part). Here, the DM can look up what to do if a specific instruction name occurs in a QSpec.

All instructions can either be executed for **interests** or for **dislikes**. For the latter, the instruction name indicates that it is an instruction for a dislike and, in addition, the attribute `dislike` is set in the node by executing the instruction.

Instruction Types

We distinguish **four instruction types** that can occur in a QSpec:

Type 1: “create a new node (with yet *unknown* attribute settings)”
Instruction name: “new_node”

This instruction is set in a QSpec for which an answer of type “**free user answer**” is expected. The DM chooses the instruction processing rule that matches this instruction name and performs the following actions (stated in the body of the rule):

- Call LA for answer a_i and get back the new profile node it has created.
- Call PM for inserting this new node into the profile.

Example 5.17

The last question q_i was “Bitte geben Sie noch ein Interesse ein! (Please enter yet another interest!)”. In $QSpec_i$ the expected answer type is “free user answer” and the instruction name is “new_node”. Assume that the user has answered “guter Wein (good wine)” in a_i . The DM looks up the instruction processing rule which matches the instruction name “new_node” and triggers the following actions stated in the body of the rule:

- The LA analyzes the answer “guter Wein” and creates an interest node for it with `object = “Wein”, property = “guter”`.
- The PM inserts the new interest node into the profile.

Type 2: “create a new node with already *known* attribute settings”
Instruction name: “new_node_with_<attribute(s)+value(s)>”

This instruction is set in a QSpec for which an answer of type “**ja/nein (yes/no) answer**” or “**selected value answer**” is expected. The DM chooses the instruction processing rule that matches this instruction name and performs the following actions (stated in the body of the rule):

- No LA needs to be performed here, because we already know how to relate the attribute values given in the answer to specific attributes (because we have asked for these attributes). Hence, the DM composes a new profile node itself, with the attribute-value-pairs mentioned in the instruction name.
- The PM inserts this new node into the profile.

For composing a profile node itself, the DM makes use of the profile-TG-node pair that is also stored in $QSpec_i$ (cf. Figure 5.17, Section 5.4.2.1). Remember that this node pair was selected as most suitable node pair for asking q_i about. From this node pair, the DM gets the attribute values it needs for composing the profile node. In the instruction name, it is indicated which attribute values the DM needs to get from the profile-TG-node pair. “TGNode[2]”, for example, means that the DM needs to get the `activity` value stored in field 2 of the TG node array. Various attribute combinations can be specified in the instruction name, such as

```
new_node_with_TGNode[3]
new_node_with_TGNode[0]_TGNode[2]_TGNode[6]
new_node_with_PNode[0]_TGNode[3]_dislike (indicating that this is a dislike)
etc.
```

Example 5.18

The last question q_i was “Interessieren Sie sich auch für Tauchen (Are you also interested in diving)?”. In $QSpec_i$ the expected answer type is “ja/nein (yes/no) answer”. $QSpec_i$ specifies that if the user answers “yes”, the instruction name is “new_node_with_TGNode[0]”, i.e. the node will be an interest. If the user answers “no”, the instruction in $QSpec_i$ is “new_node_with_TGNode[0]_dislike”, i.e. the node will be a dislike. The DM checks the answer, looks up the instruction processing rule which matches the instruction name, and executes the following actions:

- The DM composes a new node itself: either an interest with `object = “Tauchen”` or a dislike with `object = “Tauchen”` and `dislike = “Tauchen”`.¹²
- The PM inserts this new node into the profile.

Type 3: “add yet unknown attribute values to existing node”

Instruction name: “add_<attribute>”

This instruction is set in a $QSpec$ for which an answer of type “free user answer” is expected. An existing profile node is supposed to be updated. The DM chooses the instruction processing rule that matches this instruction name and performs the following actions (stated in the body of the rule):

- Call LA for answer a_i and get back the attribute-value pairs it has created from the answer and which it passes back in the form of an interest node.
- Call PM for inserting the new attribute-value pairs into the existing profile node. This is accomplished by comparing the new interest node, which the LA has passed back, with the existing profile node and by updating the changes.

Example 5.19

The last question q_i was “Welche Aktivität verbinden Sie mit Musik? (Which activity do you associate with music)?”. In $QSpec_i$ the expected answer type is “free user answer” and the instruction name is “add_activity”. Assume that the user has answered “regelmäßig machen (doing, regularly)” in a_i . The DM looks up the instruction processing rule which matches the instruction name “add_activity” and triggers the following actions stated in the body of the rule:

- The LA analyzes the answer “regelmäßig machen” and creates an interest node for it with `activity = “machen”` and `frequency = “regelmäßig”`.
- The PM compares this new interest node with the existing profile node and inserts the new information into the profile node.

In Example 5.19, we can see that with the call to the LA, we can actually capture all additional information given by the user, not only the `activity` value. The name “add_activity” merely indicates the attribute value we may expect in any case here.

¹² The object value is repeated for the value of the dislike attribute.

Type 4: “add values to already known attributes of an existing node”

Instruction name: “add_<attribute(s)+value(s)>”

This instruction is set in a QSpec for which an answer of type “**ja/nein (yes/no) answer**” or “**selected value answer**” is expected. An existing profile node is supposed to be updated. The DM chooses the instruction processing rule that matches this instruction name and performs the following actions (stated in the body of the rule):

- No LA needs to be performed here, we already know how to relate the attribute values given in the answer to specific attributes. Hence the DM adds to the existing profile node the attribute-value pairs mentioned in the instruction name.
- The PM compares the existing profile node with the node composed by the DM and updates the existing profile node.

Here, also various attribute combinations are possible in the instruction name. The attribute values are also taken from the profile-TG-node pair contained in the QSpec, as described for Type 2 above.

Example 5.20

The last question q_i was “Zum Thema Musik: Interessiert Sie speziell klassische Musik? (With respect to music: Are you especially interested in classical music?)”. In $QSpec_i$ the expected answer type is “ja/nein (yes/no) answer”. $QSpec_i$ specifies that if the user answers “yes”, the instruction name is “add_TGNode[5]”, i.e. the node will be updated with an interest attribute (`property`). If the user answers “no”, the instruction in $QSpec_i$ is “add_TGNode[5]_dislike”, i.e. the node will be updated with a `dislike` attribute indicating some dislike of the user. The DM checks the answer and looks up the instruction processing rule which matches the instruction name, and executes the following actions:

- The DM either builds an interest node with all information contained in the existing profile node plus `property = “klassische”` or it builds a dislike node with all information contained in the existing profile node plus `property = “klassische”` and `dislike = “klassische”`.
- The PM adds the new information to the existing profile node about “music”.

5.4.2.4 Question Selection and Creation of Question Specifications

In this section, we will explain Task 4, which is the main task of the DM, in more detail. The main task of the DM is the selection of the contents for the next question, i.e. the DM has to determine what the next question is about. This complex process is implemented in the **Question Selection** class, which is assisted by the **Attribute Comparison** class (cf. Figure 5.16, Section 5.4.2.1).

The main method in the Question Selection class that is responsible for controlling this process is called `selectNextQuestion()`. It takes as input the Matching Result (cf. Section 5.3.2.1) created by the TGMAT. Intuitively speaking, its task is to select a set of questions that are potentially suitable for being asked in the next dialog unit and to store them in the order of their suitability in a **question queue**. From this queue, the most suitable question (that has not been used before) is selected and asked to the user. If the most suitable question has been used in the dialog before and must therefore not be applied a second time, the second-best suitable question is taken from the queue, etc. After an unused question has been found, the other questions in the queue are discarded. The succeeding dialog unit starts with

an empty question queue and a new set of questions will be selected and stored in the queue. For this reason, the questions in the queue are always very closely coordinated with the dialog situation at hand. The number of questions contained in the question queue is configurable and is set to ten at the moment.

Technically speaking, the DM does not select and store questions, but **ISpecs** that consist of **QSpecs** (cf. Section 5.4.2.1). Each QSpec represents information needed to generate a question and to react to its answer. ISpecs bundle at most two QSpecs of questions that belong together more closely in the discourse than other questions (cf. Section 5.4.2.2 “Task 6”).

If possible, the method `selectNextQuestion()` creates as many ISpecs as fit into the question queue (i.e. ten at the moment) and returns the question queue (i.e. technically speaking, a vector of ISpecs) to the DM main method. If not enough suitable node pairs can be found, the question queue may also contain less than the maximum number of ISpecs.

Our approach to selecting the contents of questions takes into account the following aspects, the influence of which will be explained in the remainder of Section 5.4.2.4:

- The similarity relations between profile nodes and TG nodes
- Application-related nodes that should be asked about with higher priority
- The weights of the TG nodes in the TGMOD
- The comparison of individual attribute settings between profile node and TG node
- Different types of fallback questions that can be applied in different dialog situations
- The role that interests and dislikes of the user play for the dialog

At this point, a remark should be made about nodes and attributes with high application priority that can be specified in the configuration settings by the application designer. With our concept and implementation, we demonstrate that the idea of asking about **nodes** with high application priority preferentially can be practically realized. QSpecs for asking about **attributes** with high application priority have been theoretically provided in our concept, but have not yet been elaborated in detail. However, we have already realized a framework that allows an easy integration of them in parallel to the (already implemented) nodes with application priority.

Now that we have given an overview of the main method of the Question Selection class, we will explain how the questions are selected in full detail. The remainder of this section will be divided into the part accomplished by the Question Selection class (“Finding Suitable Questions for the Question Queue”), and the part accomplished by the Attribute Comparison class (“Comparing Attribute Values in ‘Complete Match’-Nodes”).

Finding Suitable Questions for the Question Queue

We will begin with the tasks accomplished by the **Question Selection** class. Remember that the **Matching Result** is a vector that contains all most similar TGs that have been detected by the TGMAT. The first task of the question selection main method is to check which of the following cases applies with respect to the Matching Result:

- **Case 1: Matching Result contains exactly one most-similar TG.**
Here, the question selection operates on this single most-similar TG in the following (i.e. on all profile-TG-node pairs that have been created by the TGMAT for this TG).
- **Case 2: Matching Result contains several most-similar TGs.**
Here, the question selection further operates on all TGs in the Matching Result.

- **Case 3: Matching Result contains no most-similar TG.**

Here, the question selection operates on the first TG of the TGMOD. We chose the first TG arbitrarily, because if there is no most-similar TG, then all TGs are equally suitable or unsuitable.

The question selection chooses the adequate case and performs **three main steps** for it. These main steps are the same for each of the three cases (the distinction of the three cases only plays a rule later on, cf. “Step 2: Selecting the Most Suitable Node Pairs from all Interest Nodes” below):

1. Select **all interest node pairs** from the TG(s) (i.e. ignore dislikes).
2. From the interests, select **suitable node pairs** for the question queue.
3. Consult the **Question Selection Schema** for building one or two QSpecs for each node pair. Summarize the QSpecs for each node pair in an ISpec. Store all ISpecs in the final **question queue**.

In the following, we explain the three main steps in detail.

Step1: Selecting all Interest Node Pairs

The main purpose of this step is to create a pool of node pairs that constitutes an adequate basis for selecting a most suitable node pair for asking the next question about. We use this step to filter out node pairs we do not want to be part of this pool, such as those representing dislikes of the user.

Technically, the first step consists of the extraction of all profile-TG-node pairs that contain **interest profile nodes** from the TG(s) (i.e. the TGs we work with according to Case 1, 2, or 3 above). Node pairs that contain **dislike profile nodes** are stored properly in the profile but **are ignored** during the question selection. This is because we did not want to spend precious dialog time on dislikes of the user and their (presumably less desirable) details. We wanted to keep the number of questions as small as possible and to stick to topics that are interesting for users in order not to overstrain them.

NOT FOUND nodes in the profile that represents user interests, are not included in this selection process at the moment, i.e. no complete concept for asking questions about them has been developed yet. However, it would not be difficult to extend our approach to take into account questions that elicit further information about “NOT FOUND interests”. This information might then help to insert them into the profile graph correctly. Questions about NOT FOUND nodes could, for example, be asked alternately with or instead of fallback questions (see below). Questions about NOT FOUND nodes would mainly aim at “filling” the attributes object, activity, and favorite_items of a NOT FOUND node, because these are needed for searching a node in GermaNet (cf. Section 5.2.2.2).

A special case, which occurs very rarely in practice, is given when **no interests at all** are contained in the profile as yet, because the user has mentioned only dislikes in the dialog so far. In this case, a specific set of **fallback questions**, tailored to this situation, are added to the question queue. Some examples of fallback questions we use for this purpose are the following (their formulation can be changed if desired for a specific application):

- Wir wissen noch recht wenig über Ihre Interessen. Verraten Sie uns ein paar?
(We do not know much about your interests so far. Would you like to tell us a few?)
- Womit beschäftigen Sie sich am liebsten in Ihrer Freizeit?
(What is your favorite leisure activity?)

- Welche Dinge, Themen oder Aktivitäten interessieren Sie am meisten?
(Which things, topics, or activities are you interested in most?)

Step2: Selecting the Most Suitable Node pairs from all Interest Nodes

The idea of Step 2 is to select from the pool of interest nodes (created in Step 1) a set of most suitable node pairs and order them according to their suitability for asking the next question about. The number of selected node pairs depends on the (configurable) question queue size (cf. beginning of Section 5.4.2.4). The selected node pairs are temporarily stored in a vector that has the same size as the final question queue.

For this task, Step 2 performs **several actions in a specific order**. The order of these actions and also some action types are different for Cases 1, 2, and 3, which distinguish the Matching Result types (see above). We have determined the current order of the actions empirically by conducting many dialog tests and also added further actions over time for fine-tuning the course of the dialog. The actions are described below.

The **strategy for asking questions**, as described in Section 4.2.1.2, can be influenced by rearranging the order of actions in Step 2. This requires in-depth knowledge of the DM component, but is possible in general. With reordering these actions, one can, for instance, determine whether the dialog asks deepening questions about each interest first or whether it tries to extend the number of interests in the profile by asking broadening questions first (cf. Section 4.2.1.2). The first variant is perceived by users as “sticking to one topic longer”, whereas for the second variant topic changes occur more often.

In Section 4.2.1.2, we have explained that the **current strategy** taken in LINGUINI is to **explore related topics first (via broadening questions)**, before asking deepening questions about known interests. Although this comes with the disadvantage that the topics are changed more frequently and known topics are sometimes only recaptured at a later point of the dialog, the advantage of this strategy is that the system can suggest more new topics and can check whether the user finds these interesting too. As a consequence, the dialog is more dynamic, diversified, and interesting than if deepening questions were asked about each topic successively, which some users perceive as rather boring.

The **actions** performed in Step 2 for the three cases are listed here in their current order in LINGUINI:

Case 1 (one most-similar TG):

Actions performed in Step 2:

1. Select from the pool of interest node pairs all pairs that contain **profile nodes with high application priority** and store them in a vector.
2. Select from the rest pairs in the pool all **node pairs with similarity relation “hyponym”** and add them to the vector.
3. Select from the rest pairs in the pool all **node pairs with similarity relation “siblings”** and add them to the vector.
4. Select from the rest pairs in the pool all **node pairs with similarity relation “hyponym”** and add them to the vector.
5. Select from the rest pairs in the pool all **node pairs with similarity relation “complete_match”** and add them to the vector.
6. In case the vector still contains less elements than the question queue size, fill it up with **fallback questions**.

The first action makes sure that **nodes with high application** priority are preferred for asking questions about. In Case 1, a similarity between the profile and one of the TGs has

been detected by the TGMAT. Therefore, the **similarity relations** between nodes are taken into account in the Actions 2-5. The questions are ordered in such a way that we ask about similarity relations with a rather low relation value (cf. Section 5.3.2.4) first (Actions 2 and 3) and then ask about those with a higher relation value (Actions 4 and 5). This corresponds to our dialog strategy of asking broadening questions first and deepening questions later. With Action 6, we fill the rest of the question queue with fallback questions¹³. This happens if the intended question queue size is not yet reached, but no node pairs with a similarity relation other than “no mach” are left that could be used for the queue.

Case 2 (several most-similar TGs):

The actions performed here are the same as for Case 1, with the following modification: After node pairs with similarity relation `complete_match` are selected (Action 5), the next step is to

- select from the rest pairs in the pool **the first node pair** which contains the **TG node with the highest weight**.

After this, Action 6 is performed as in Case 1. This order has been chosen, because we first want to exploit again the **similarity relations** between profile nodes and TG nodes (as in Case 1). The new action is inserted afterwards, because we want to ask about the **TG node with the highest weight**, i.e. the highest discriminating power in Schulze’s model (cf. Section 5.3.1.2). TG nodes with high weights represent topics that have a high power for splitting users into different groups. By answering a question about a TG node with a high weight, the probability increases that the similarity between the profile and one of the TGs will converge in the next TGMAT. Therefore, this action is performed in Case 2, where the TGMAT is “undecided” between several most-similar TGs. Only after this action, the rest of the vector is filled up with fallback questions.

Case 3 (profile is not similar to any TG):

Actions performed in Step 2:

1. Select from the pool of interest node pairs **the first node pair** which contains the **TG node with the highest weight** and store it in a vector.
2. Select from the rest pairs in the pool all pairs that contain **profile nodes with high application priority** and add them to the vector.
3. Select from the rest pairs in the pool all **node pairs with similarity relation “related_nodes”** (i.e. profile node and TG node are contained in the same GermaNet subtree with depth 2).
4. In case the vector still contains less elements than the question queue size, fill it up with **fallback questions**.

If the Matching Result has only achieved Case 3, i.e. the profile is not similar to any of the TGs as yet, the most important task is to achieve Case 2 or 1 in the next TGMAT by asking the next question about a highly discriminating attribute. Therefore, we ask about a **TG node with high weight** first (Action 1) in order to increase the probability of a future convergence between the profile and one of the TGs. With Action 2, we take into account the nodes with **high application priority**, as in the other cases. We do not have similarity relations between profile and TG nodes, because there is no similarity of the profile with one of the TGs, hence

¹³ Fallback questions will be described in “Step 3: Consulting the Question Selection Schema for Building Question Specifications” below.

we cannot use these for asking questions. Instead, we chose the solution to investigate the semantic environment of a profile node here, i.e. we take the grandparent of the profile node and consider the entire tree under this grandparent. In case a TG node is contained in this subtree, the node pair is selected (Action 3). After this, the rest of the vector is filled with fallback questions as for the other two cases (Action 4).

Step3: Consulting the Question Selection Schema for Building Question Specifications

The main purpose of Step 3 is to take the vector of node pairs created in Step 2 and transform it into the final question queue, which contains all necessary information for constructing natural language questions about the node pairs.

For this task, we proceed as shown in Figure 5.18. The **vector of selected node pairs** contains as many elements as the **question queue** is supposed to contain finally (ten in our current configuration). For each of the ten node pairs, one or two QSpecs are generated and stored in an ISpec. Hence, we finally have ten ISpecs that are inserted into the question queue. Then the question queue can be returned to the DM main method.

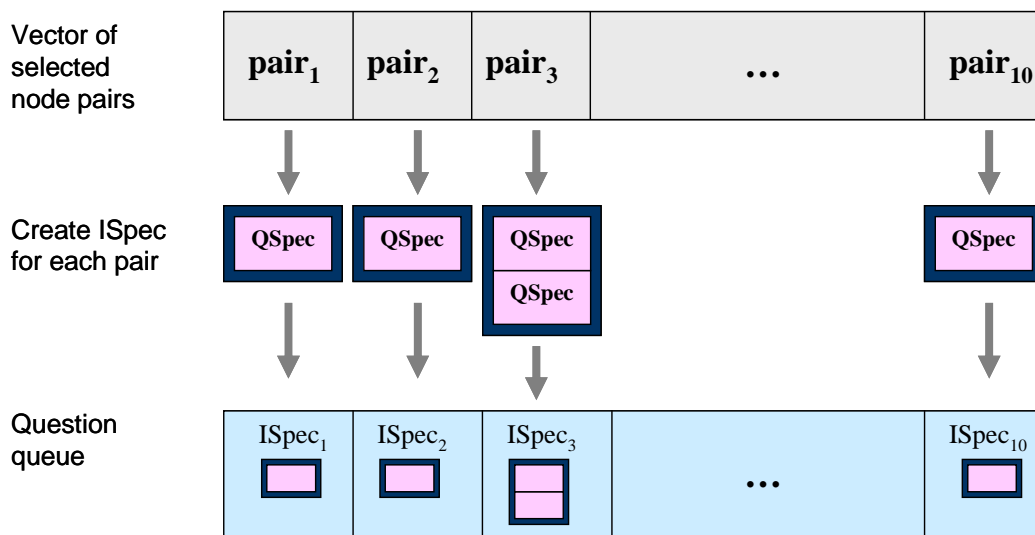


Figure 5.18: Constructing ISpecs for node pairs and inserting them into the question queue

In order to construct an ISpec for a node pair, the Question Selection class uses the **Question Selection Schema**. Intuitively speaking, the Question Selection Schema defines what kind of question makes sense considering the specific properties of a node pair. It consists of 11 **conditional rules**. These rules are used to check the properties of the nodes contained in the node pair and to decide on this basis which information should be contained in the corresponding ISpec. The conditional part of a rule therefore accomplishes the checking of specific node properties, whereas the body of the rule determines how the corresponding ISpec has to be constructed. Example 5.21 shows a sample rule, in which an ISpec that contains one QSpec is constructed. An example for a rule constructing an ISpec with two QSpecs is shown below, in Example 5.23 in the section “Comparing Attribute Values in ‘Complete Match’-Nodes”.

Example 5.21

Hyponym Rule: <profile node is hyponym of TG node >

Example: Profile: "Segeln (sailing)"; TG: "Wassersport (water sports)"

If the following condition is met, the rule is applied ...

if similarity relation of profile node and TG node = hyponym

... and the body of the rule is executed, i.e. a QSpec is built:

then build a QSpec by adding the following information to it:

set precondition = empty

store profile node in the QSpec

store TG node in the QSpec

store profile node ID in the QSpec

set question frame, answer type, and instructions:

Here, we distinguish two cases: either an object (which is not "--") is set in both nodes, or an activity is set in both nodes:

if object is set in profile node and in TG node

then question frame = "Da Sie pObject erwähnt haben, interessieren Sie sich auch für tgObject allgemein?"

set answer type = yes/no

set instructions for answer processing:

if answer = „ja“

then instruction = new_node_with_TGNode[0]
(create a new profile node with the object value of the TG node; this node represents an interest of the user)

else if answer = „nein“

then instruction =
new_node_with_TGNode[0]_dislike
(create new profile node with object value of the TG node as dislike value; this node represents a dislike)

end if

else if object is not set, but activity is set in profile node and in TG node:

then question frame = „Da Sie pActivity erwähnt haben, interessieren Sie sich auch für tgActivity allgemein?“

set answer type = yes/no

set instructions for answer processing:

if answer = „ja“

then instruction = new_node_with_TGNode[2]
(create a new profile node with the activity value of the TG node; this node represents an interest of the user)

else if answer = „nein“

then instruction =
new_node_with_TGNode[2]_dislike
(create new profile node with activity value of the TG node as dislike value; this node represents a dislike)

end if

end if

add the created QSpec to the ISpec

end if

return ISpec

The properties that can be checked in the **conditional part** of the rules are the following:

- Does the node pair contain a TG node with highest weight?
- Does the pair have one of the similarity relations (*hyperonym*, *siblings*, etc.) or is it a *no_match* pair, i.e. profile and TG node are not semantically related?
- Do specific attribute settings occur in the profile node and/or the TG node?

In the **body** of a rule, one or two QSpecs are constructed for a node pair that fulfills the conditions of the rule. The QSpecs constructed for a node pair are stored in an ISpec, which is added to the question queue. When all elements have been processed, the question queue is returned to the DM main method.

In Step 2 above, we have explained that the DM has the possibility of filling up the question queue with QSpecs for so-called **fallback questions**. The DM makes use of this possibility if there are only node pairs with similarity relation *no_match* left in the pool of interest nodes. Fallback questions are also relevant for Case 3, where the profile does not show a similarity to one of the TGs at all. In these cases, the DM has to find another way to choose a next question. It has a contingent of fallback questions which it can use for this purpose. At the moment, we have restricted this contingent to five fallback questions, because if the DM cannot create questions adaptively to user interests any more, it is better to stop the dialog before asking too many very general fallback questions. This view is supported by our final user study (cf. Section 6.3), in which users confirmed that too many fallback questions are in fact not appreciated very much.

The system employs two types of fallback questions. First, the fallback questions we use in case the **profile does not contain any interest nodes** as yet (but perhaps some dislikes) (cf. Section 5.4.2.4 “Step 1”). This case occurs very rarely. More often, the system makes use of the second type of fallback questions for filling up the question queue in case the **profile does already contain interest nodes** but these do not show a similarity with the TGMOD (i.e. the profile-TG-node pairs in which they occur have the similarity relation *no_match*). Examples for such fallback questions are:

- Welche Dinge, Themen oder Aktivitäten interessieren Sie noch?
(Which other things, topics, or activities are you interested in?)
- Bitte geben Sie noch ein Interesse ein!
(Please enter one more interest!)
- Beschreiben Sie bitte 2-3 Interessen, denen Sie vorwiegend außer Haus nachgehen!
(Please describe 2-3 interests you pursue outside your home!)

Again, the formulation of the fallback questions can be changed easily if this is desired by some application. The QSpecs of fallback questions are constructed in the same way as all other QSpecs: conditional rules – as the one shown in Example 5.21, but less complex – contained in the Question Selection Schema are used for constructing the QSpec of a fallback question and for adding it to the ISpec.

A special case, which is handled differently from the other cases, is the similarity relation *complete_match*. For node pairs with this similarity relation, the check of the various combinations of attribute settings is so complex that this is performed by a separate class, the **Attribute Comparison**. Hence, if the Question Selection Schema detects a “*complete_match*” node pair, it does not construct the QSpec itself, but delegates this task to the Attribute Comparison class. For all other similarity relations, the conditional rules of the

Question Selection Schema construct the QSpecs themselves. The functioning of this attribute comparison process will be explained in the next section.

Comparing Attribute Values in “Complete Match”-Nodes

Finally, we will describe the tasks accomplished by the **Attribute Comparison** class. If the Question Selection Schema detects a node pair with similarity relation `complete_match`, it delegates the construction of the QSpec to the class Attribute Comparison. For these node pairs, the attribute settings check is more complicated because the exact formulation of a question frame depends very much on the presence or absence of specific attribute values in the profile node and/or the TG node.

Questions about `complete_match` node pairs are always **deepening questions**, i.e. the system tries to find out details about an already known user interest that is also contained in one of the TGs. In an early stage of the dialog, it occurs very often that a TG node in a `complete_match` node pair contains more information than the corresponding profile node. The Attribute Comparison class tries to make use of this additional information contained in the TG nodes in order to ask questions more “intelligently”. In other words, it tries to get clues from the corresponding TG node for eliciting further information about a user interest, as illustrated in Example 5.22:

Example 5.22

Assume that in a given `complete_match` node pair the user interest is “Brettspiele (board games)” and the TG interest is “Brettspiele, speziell Monopoly (board games, especially Monopoly)”. The TG node is represented as `<object = “Brettspiele”, favorite_items = “Monopoly”>`. Here, the TG node provides the clue that it would make sense to ask about the `favorite_items` value of this user interest. As a consequence, the system could ask whether the user likes to play Monopoly, or in case not, what else she likes to play.

The Attribute Comparison class contains 31 conditional rules that are formally of the same kind as the rules in the Question Selection Schema. A sample attribute comparison rule is shown in Example 5.23. It constructs two QSpecs for a given node pair:

Example 5.23

Attribute Comparison Rule 1.8: <object is set; ask for favorite_items>

Example: Profile: "Musik"; TG: "Musik, speziell Mozart (music, especially Mozart)"

If the following condition is met, the rule is applied ...

```
if object in profile node is identical with object in TG node
  AND favorite_items is set in TG node but not in profile node
```

... and the body of the rule is executed, i.e. two QSpecs are built:

```
then build QSpec1 by adding the following information to it:
```

```
  set precondition = empty
  store profile node in QSpec1
  store TG node in QSpec1
  store profile node ID in QSpec1
```

```
  set question frame:
```

```
    if activity is set in profile node
      then question frame = „A propos pObject pActivity:
        Mögen Sie tgFavoriteItems?“
```

```
    else if activity is not set in profile node
      then question frame = „A propos pObject: Mögen Sie
        tgFavoriteItems?“
    end if
```

```
  set answer type = yes/no
```

```
  set instructions for answer processing:
```

```
    if answer = „ja“
      then instruction = add_TGNode[9]
      (add favorite_items value of TG node to profile node)
```

```
    else if answer = „nein“
      then instruction = add_TGNode[9]_dislike
      (add favorite_items in TG as value of dislike attribute to profile node)
    end if
```

```
  add the created QSpec1 to the ISpec
```

```
next build QSpec2 by adding the following information to it:
```

```
  set precondition = „nein“
  store profile node in QSpec2
  store TG node in QSpec2
  store profile node ID in QSpec2
```

```
  set question frame:
```

```
    question frame = "Welche/s/n Lieblings-pObject haben
      Sie (dann)?"
```

```
  set answer type = free user answer
```

```
  set instructions for answer processing:
```

```
    instruction = add_favorite_items
    (analyze the answer with the LA and add the extracted favorite_items
      value and possible further information to the profile node)
```

```
  add the created QSpec2 to the ISpec
```

```
end if
```

```
return ISpec (consisting of QSpec1 and QSpec 2)
```

If the system detects a `complete_match` node pair, it could theoretically ask about all attributes that are allowed in an interest node, i.e. locations, ratings, frequencies, participants, etc. However, asking about all of these attributes for every user interest sequentially may

cause the dialog to become rather rigid and boring. Therefore we have integrated two mechanisms to avoid rigidity:

- A **configurable threshold** determines the maximum number of deepening questions asked for one `complete_match` pair. At the moment it is set to three in our prototype. If the number is increased, more questions are asked about one topic. If the number is decreased, the user perceives that topic changes occur more frequently.
- Moreover, the (currently three) deepening questions asked for each `complete_match` pair are **selected randomly**. With this method, we can avoid, that for each interest the user is asked questions about the same three attributes. So, for the interest “swimming”, the user might be asked whether she swims frequently, where she swims, and if there are other participants. For the interest “theater”, she might be asked how important this is for her, which activity she performs with respect to this interest, and whether there are further participants.

Finally, we would like to mention that our approach to deepening questions does not yet integrate **pragmatical information**, except for the one that can be derived from the TGMOD (cf. Example 5.22 above). It may be theoretically correct, for instance, to ask about participants for the interest “reading the newspaper”, but pragmatically this does not make much sense, because most users prefer to read the newspaper alone. Also asking about a location can be pragmatically less relevant for some interests. However, there is also the opposite case, where users found it inspiring and interesting that the system asked them about a location in connection with one of their interests (cf. Section 6.3.1.5 “1.1 Emotions”). Since the cases in which questions about specific attributes do or do not make sense cannot be generalized easily and the semantics of each individual interest would have to be analyzed in detail, to develop a solution for this problem would have exceeded the scope of this thesis. It would be interesting to investigate this topic in the future, however.

5.5 Summary

This chapter presented the main system components of LINGUINI, the design of which and their way of interacting constitutes the main contribution of the thesis, together with the Profile Model presented in Chapter 4. Together, these components contribute to an integrated profile management and dialog management framework. For the design of these components, we particularly considered the requirements we have formulated for this framework in Section 1.3.

The main contributions of the individual components to this framework are the following. The **Language Analysis** component contributes both to the dialog part by enabling the interpretation of explicitly entered natural language user interests and to the profile part by formally representing the information extracted in form of complex interest nodes. The contribution of the **Profile Manager** is to find semantic relations between the information collected by the Language Analysis and the information stored both in the user profile and GermaNet. Together with the Profile Model, the Profile Manager constitutes the central part of the profile management. The **Target Group Matching** is an important functionality that provides the basis for the question selection performed afterwards by the Dialog Manager. It mainly contributes to the dialog management by preselecting a pool of profile and target group node pairs that are most suitable for asking the next question about. For this purposes it uses an extended sociological **Target Group Model** that represents groups of users according to their interests. The information stored in the Target Group Model further helps the Dialog Manager

to ask questions more “intelligently”, e.g. by providing clues as to whether it makes sense to ask for favorite items, locations, etc. in relation to a specific interest the user has mentioned. The core of the dialog part is the **Dialog Manager**, which is responsible for controlling the interaction with the user. Moreover, it also controls the central internal processes of the system. Its main task is the selection of suitable questions which adapt to what the user has already mentioned.

6 Implementation and Evaluation

After having presented the concept for our profiling approach in the previous chapters, we will now explain how our prototype system has been implemented and how we evaluated it. In Section 6.1, we describe a user study we conducted prior to the design and development of LINGUINI. It investigates how users formulate their personal interests in natural language. The design and development of the Profile Model and the LA component were strongly influenced by the results of this study. Section 6.2 is dedicated to implementation aspects. It describes in what respects the implementation of the prototype deviates from the concept presented in Chapters 4 and 5. In Section 6.3, we will describe the final evaluation of our prototype system and discuss its results. For this evaluation we conducted a user study about dialog adaptability, user acceptance, and development status of the prototype, which was followed by a profile quality assessment.

6.1 Empirical User Study about Preferred User Input

6.1.1 Type and Aims of the User Study

Section 6.1 describes the user study we conducted *prior* to the design and development of our system. The **aim of this user study** was to find out, how users would formulate their personal interests in natural language, if no or only few restrictions were placed on them. For the acquisition of profile information, the cooperation of the user is essential. Therefore, a dialog profiling system should allow users to enter information in the way they prefer most. This means in particular, not to force them to enter longer input than they consider natural in this situation and not to restrict them to single words in case they want to provide more information about a specific interest.

Another reason for conducting this study was that we needed to know which level of input complexity we have to process with our system components. The preferred input length and also its grammatical complexity, for example, has a direct impact on the **Language Analysis** component. For longer input (e.g. long sentences or texts), the correlations between grammatical components in the input may become more difficult to analyze. For very short input (e.g. words or short phrases), the LA may not find enough contextual information in order to interpret the input correctly. Different NLP tools can be applied according to the type of input expected in the profiling system. Furthermore, the development of a suitable formal representation of user interests, i.e. the **Profile Model**, also strongly depends on the information contents and structure of the input.

Since we did not have a speech interface at our disposal, the data in this study were acquired by means of typed user input. We investigated data collected in **three different scenarios**. In the **first scenario**, we collected user data ourselves by asking subjects to send their interests by email. They were asked to formulate their interests as a list of natural language expressions (headwords), in which the interests were separated by a semicolon. We did not make any restrictions as to the contents or the syntactic structure of these expressions, though. The emails were processed anonymously. 47 users of different age and professional back-

ground took part. This very early study was conducted for English user input because at the beginning of the project we had not yet decided whether the prototype would be developed for German or for English.

We complemented the data obtained in the first scenario with data collected by other projects in **two further scenarios**. These other scenarios comprised personal interests formulated in German and entered

1. by visitors of a German conference during registration¹⁴
2. in a Web forum by members of a German Techno community¹⁵

In both cases, no instructions on how to formulate or enter the interests were given. For this thesis, we linguistically analyzed the collected user interests from all three scenarios manually in order to decide on the type of LA and NLP tools needed for LINGUINI and in order to develop a formal representation of user interests.

6.1.2 Results

We found that, in all three scenarios most users entered phrases (without subordinate clauses) instead of entire sentences. The phrases they entered were rather short and of simple syntactic structure. Figure 6.1 shows some examples of user input taken randomly from the three data sets:

<p>Scenario “Email User Study”:</p> <p>reading books listening to bach and beethoven esp piano music kung fu sailing meeting friends especially oriental countries good wine</p> <p>Scenario “Conference Registration”:</p> <p>Logik und Mathematik Tango Argentino felsklettern visuell schöne Gegenstände Entwicklung von Galaxien Rad fahren</p> <p>Scenario “Techno Community”:</p> <p>ganz viel Sport und Musik schwimmen Freunde treffen Electro bevorzugt den Osten erkunden Progressive Trance spielen mit meiner Tochter</p>
--

Figure 6.1: Samples of personal interests entered by users freely in different scenarios

¹⁴ These data were collected for the Meeting Mirror project (www.cobricks.de//mm/) at the Technical University of Munich.

¹⁵ The Techno community data were collected within the framework of the work conducted by [Groh 2005].

The data show a high degree of homogeneity with respect to the syntactic structure of the input. Most users (with only a few exceptions) typically entered the following input types:

- More or less complex noun phrases (e.g. “Autos (cars)”, “Tango Argentino”, “visuell schöne Gegenstände (visually beautiful objects)”)
- Constructions containing a verb and an object and/or various types of verb modifiers (e.g. “Freunde treffen (meeting friends)”, “spielen mit meiner Tochter (playing with my daughter)”, “öfters tanzen (frequently dancing)”)
- Sometimes, ratings were added to the interests (e.g. “especially oriental countries”)

6.1.3 Discussion

The homogeneity of the data collected in the three scenarios clearly shows that

- most users prefer a rather **short formulation** for their interests (typically 1-5 words)
- the interests are of a rather **simple syntactic structure** (i.e. more or less complex phrases without subclauses)

From these results, we conclude that – at least for typed input – a **reduced effort** seems to be important for users. This correlates with the results of an evaluation conducted by [Chai et al. 2001]. They evaluated their Web-based natural language dialog system, that helps users find information in the E-Commerce context. Their evaluation showed that users of dialog-based E-Commerce platforms prefer short input of rather simple linguistic structure.

This observation had several implications for the development of our profiling system. The given simplicity of the input suggests the use of **shallow parsing techniques** for the linguistic analysis. We therefore decided to restrict the allowed user input to phrases like the ones that occurred in the user study and to employ the NLP tool TreeTagger and its associated shallow parser for analyzing the phrases. Our study shows that this restriction of the input is not forced upon users, but correlates with their own preferences.

The results of the study also influenced our **Profile Model**. From the linguistic analysis of the interests – and especially from the types of verb modifiers and additional expressions (e.g. ratings) that occurred in user interests – we derived the complex semantic attribute-value structure of interest nodes in our profile graph (cf. Section 4.4.2.1).

Another observation can be made with respect to the **language** chosen for interacting with the user. The similarity of the interests with respect to syntactic structure in English and in German suggests that our approach is theoretically applicable for both languages equally well and also for further languages that have a related grammatical structure.

The results of the study refer to **typed user input**. It was not possible within the scope of this thesis to investigate whether the same input structures also occur for spoken language. Since the same results occurred independently in very different scenarios for different types of users, however, it seems likely that they will also hold for speech applications.

6.2 Implementation of the LINGUINI Prototype System

With the implementation of the LINGUINI prototype system, we want to show that the solution developed for the two main aims of this thesis can in fact be realized technically:

- The first main aim of this thesis was the development of an **adaptive natural language profiling dialog** that reacts adaptively to the interests the user has mentioned and that is capable of handling not predefined user input. For evaluating this aspect, we needed to implement a system framework which allows conducting a profiling dialog with the system. The components that had to be realized for this framework were the Dialog Manager, the Target Group Matching and Target Group Model, the Language Analysis, and the User Interface.
- The second aim was to develop a **profile management approach** and to combine it with the above described dialog approach. This profile management approach has to allow the explicit acquisition and maintenance of user information and the representation of user interests and semantic relations between them in a structured way. The system components we implemented in order to evaluate this part were the Profile Manager and the Profile Model.

Moreover, we wanted to evaluate the prototype implementation both with a **user study** in order to assess the achieved level of dialog adaptability and user acceptance and with an objective profile quality assessment of the profiles created in this user study.

The combination of the two aims described above in connection with the intended user study argued for the realization of the complete system (i.e. all main components), rather than for a partial implementation of only some components for illustrative purposes. As a consequence, we have implemented the main functionality of all central system components exactly as described in Chapters 4 and 5. Here in Section 6.2, we therefore only describe deviations from and extensions of the concept for technical or practical reasons.

6.2.1 The Project Framework

In order to test LINGUINI, we embedded it into the product specification and recommendation platform for individualized products developed in the SFB 582 research project (www.sfb582.de).¹⁶ The specification and recommendation platform is described in [Leckner et al. 2003, Stegmann et al. 2003, Leckner et al. 2004, Renneberg et al. 2004]. The platform already offers two explicit profile acquisition methods, one for novices and the second one for experts, both, however, only based on online questionnaires the user has to fill in. The dialog profiling method of LINGUINI represents a third profiling method in this platform. Technically, LINGUINI merely uses the same Web server and the registration functionality of this platform. In all other respects it is completely independent of the SFB 582 specification and recommendation system.

6.2.2 Overview of the Prototype System

6.2.2.1 Technologies Used for Implementing the Prototype System

The Web-accessibility for our prototype system has been realized by means of the Apache Tomcat Web server technology. LINGUINI runs as the application of a Tomcat server under Linux. This is the same server as used for the SFB 582 platform (cf. Section 6.2.1). In general, however, LINGUINI does not depend on a specific operating system and can be used

¹⁶ The SFB 582 project was concerned with different aspects of the development of individualized products close to the market.

either under Linux, Unix, or Windows. The external resources and tools used in LINGUINI (i.e. the TreeTagger, chunker, and GermaNet) are available for all of these operating systems.

The LINGUINI components have been implemented in JAVA. The Web User Interface has been implemented by using the JAVA Servlet technology. For modeling user information we chose XML in order to guarantee an easy exchangeability of the user profiles between applications. Another reason for choosing XML was that GermaNet, our main source of linguistic knowledge in the system, is also represented in XML and that our user profiles are derived from GermaNet.

We use a MySQL database system for storing

- user profiles
- TGs contained in the TGMOD
- dialog configuration files
- the Dialog History of each user
- the Semantic Classes Lexicon used by the LA

6.2.2.2 Architecture and Functionality of the Prototype System

The prototype system can be used to conduct a complete profiling dialog with the system. It runs on a server and can be accessed by any Web browser on the client machine. The system architecture has been implemented as illustrated by Figure 4.3 in Section 4.3, except for some small modifications described in the remainder of Section 6.2.

For the final evaluation (cf. Section 6.3), we implemented the system in such a way that a separate user profile is created for each dialog interaction with the system. Each subject in the user study conducts exactly one dialog. For allowing dialog interactions that extend over several sessions in practical use, the prototype has to be slightly modified in the following respect. An existing profile should be loaded when the user logs into the system. For this purpose, the User ID (that belongs to the user's login name) has to be stored together with the Profile ID created during the first interaction with this user.

6.2.2.3 System Component Tests

The main system components have been tested intensely by means of separate component tests before evaluating the system in the final user study. We have tested both LA and PM with the data collected in the empirical user study at the beginning of the project (cf. Section 6.1). Furthermore, we enriched this test set with manually constructed interest phrases that show a more complex syntactic structure (e.g. nominal modifiers, all sorts of combinations of attribute types, etc.). Thus we were able to test the Attribute Extraction Rules in the LA more thoroughly. The LA has been tested with this extended data set. We found that we can handle most of the input structures satisfactorily with our nine Attribute Extraction Rules.

The PM has also been tested with this extended test set for three cases: 1) creation of a new profile, 2) updating of an existing profile with new attribute-value pairs and attribute values, and 3) relocating profile nodes that have previously been children of the NOT FOUND node (cf. Section 5.2). All cases have been tested with sets of user interests of different lengths (up to 5, up to 10, and more than 10 interests). We also used different combinations of attributes within one user interest. Attributes could have one or several values each. We also tested whether all paths contained in GermaNet for a specific interest were completely included in the profile by our search algorithm. Our tests demonstrated that these functions work properly.

The TGMAT has also been tested as an independent component with different parameter variations. The aims of the TGMAT test were, first, to ensure correct functioning and, secondly, to find out the optimal parameter settings for the TGMAT. Among other things, we tested the following main aspects. We tested the TGMAT with the “dislike parameter” (cf. Section 5.3.1.1) on and off, i.e. taking into account dislikes of the user or not. Here, we tested the cases in which an interest contained in the profile is a dislike in one of the TGs and vice versa and we also tested dislikes in profile and TGs that are independent of each other. We also varied the “Matching Result parameter” which determines how many TGs are included in the Matching Result. We found that the ideal value for this parameter is 0.0125 (cf. Section 5.3.2.4). We also tested the TGMAT with profiles that are identical to one of the TGs, with profiles that contain only some similar nodes, and with profiles that show no similarity at all to one of the TGs. The calculated Matching Degree and the Matching Result have finally been compared with a human judgment of similarity between the profiles and the TGs. We also tested whether similarity relations between nodes are determined correctly on the basis of their GermaNet paths.

Finally, the DM has been tested for various parameter settings as well. We also varied the order of actions performed during the process of selecting the most suitable node pairs (cf. Section 5.4.2.4 “Step 2”) in order to find out their impact on the dialog flow. We compared the dialog strategy of asking broadening questions first with the strategy of asking deepening questions first and decided to keep the first variant because it makes the dialog more diversified and acquires more new interests for the profile. We tested all functionalities offered by the DM, such as taking into account nodes with high application priority, the generation of fallback questions, the Dialog History, the interaction with the User Interface, etc. On the basis of these tests, we chose the DM parameter settings for the final user study (e.g. dialog length = 30 questions at most; 3 deepening questions per interest).

6.2.3 Components

The system components have been implemented according to the concept presented in Chapters 4 and 5. In this section, we therefore mainly describe deviations from this concept and extensions of the concept for technical or practical reasons.

6.2.3.1 The Language Analysis

The LA concept as presented in Section 5.1 has been implemented completely in LINGUINI. The LA has access to the external NLP tool TreeTagger and its associated chunker for syntactically analyzing the user input. For the integration of the TreeTagger, we had to solve the following problem. The TreeTagger is a compiled program, which can be accessed by means of a shell command, but which does not offer a JAVA-API. For making the TreeTagger accessible to our JAVA classes, we therefore used a wrapper developed by European Media Laboratory GmbH (EML) in Heidelberg. A wrapper is typically used to make software compatible that has been written in different programming languages or runs on different operating systems or machines. This wrapper allows us to execute the TreeTagger from within a JAVA program and returns the raw output of the tagger in a JAVA method.

The wrapper is implemented in the **TreeTagger Wrapper** class. This class is called by the **Tagger Chunker** class we have developed and which represents the link between the wrapper and the LA main class. The Tagger Chunker class receives the raw tagger output from the wrapper and returns to the LA either a tagged and chunked user answer or the lemmatized forms of words in the user input (depending on the method called in the Tagger Chunker class) (cf. Section 5.1.1.1).

6.2.3.2 The Profile Manager

Parsing and Visualizing XML User Profiles

For updating user profiles, the PM needs to read the XML tree representation (cf. Section 4.4.2.2) of an existing profile and perform various operations on it, such as inserting or deleting nodes or adding new attributes and values to a node. Technically, we achieve this by **parsing** the profile with an XML parser. We use Xerces, a widely used XML parser which is available under the Apache Software License (xerces.apache.org). Xerces allows us to parse the XML representation of a profile into a Document Object Model (DOM). The DOM-API defines a set of methods for accessing and manipulating the contents and structure of information stored in XML documents in a hierarchical manner. It allows, for instance, easily deleting or adding nodes or elements contained in nodes.

We have not yet implemented a functionality for **visualizing** an XML profile representation for the user such that it does not appear too complex. At the moment, the created profiles can only be viewed in their XML encoding by means of the database standard view.

Accessing Information Contained in GermaNet

The main problem we had to solve during the implementation of the PM was to provide access to GermaNet. The PM uses GermaNet to locate interest nodes, to extract the paths for all their meanings, and to build a profile graph that is a subgraph of GermaNet. GermaNet is represented as a large set of individual XML files storing about 62 000 lexical units (words). Unfortunately, at the beginning of our project, no JAVA-API existed for GermaNet. As a consequence, a JAVA-API has been developed in a related project at the Technical University of Munich as a student programming project [Ast 2003], but it proved not fast enough for a real time application such as dialog-based user profiling.

In the final phase of this thesis, we were able to obtain a JAVA-API for GermaNet which has been developed at European Media Laboratory GmbH (EML) [Gurevych and Niederlich 2005b]. This API proved to be much faster than the API developed by [Ast 2003], but has completely different access methods to the GermaNet information. We could then adapt our classes to cooperate with this faster API, but we had to do without integrating the word sense disambiguation component (cf. Section 3.6.3) into our prototype. This component had been implemented in a diploma thesis (cf. [Eiseler 2005]).

The problem with the disambiguation component was the following. The API developed at EML does not allow accessing the GermaNet-internal IDs of synsets, which we use as node IDs in the profile graph. These node IDs are necessary for identifying different meanings of nodes uniquely during word sense disambiguation, because node names are not necessarily unique. For this reason, the disambiguation component implemented by [Eiseler 2005] is based on the previously used GermaNet-API by [Ast 2003], which provides the unique node IDs from GermaNet, but which unfortunately was not fast enough for practical use.

At present, we deal with the missing node IDs in the prototype as follows. Since the node IDs are only used for word sense disambiguation, we could simply replace them by automatically constructed “dummy IDs” of the type “x.0.<node_name>”. These IDs are missing the hyperonym information (e.g. “nArtefakt”) and node number (e.g. 2711) contained in a GermaNet synset ID (e.g. “nArtefakt.2711.Buch”). Hence, the interest node “Buch (book)”, for example, has the dummy ID “x.0.Buch” at present in our implementation instead of the original “nArtefakt.2711.Buch”. Hence, the node IDs we currently use in the prototype are different than the node IDs used by the disambiguation component.

Of course, this is not the finally intended solution, but only in this way we were able to use the faster API for the final evaluation of the prototype. The ultimate solution, however, which also allows the integration of the disambiguation component into LINGUINI, should

be to extend the EML-API for providing the original GermaNet synset IDs. This was not possible for the evaluation as yet, because the source code of the EML-API is not freely available and EML did not have the human resources for making the changes at that point in time.

Information Missing in GermaNet

Another problem with respect to GermaNet was its smaller coverage as compared to the English WordNet (cf. Section 3.5). Since it was started later than WordNet and is still under construction, GermaNet does not yet cover all words of the German language. The problem with this for LINGUINI is that user interests which cannot be found in GermaNet are stored in the profile as “NOT FOUND nodes”, i.e. the information is not lost, but it cannot be related to other information items semantically.

Since the primary intention of GermaNet is not to cover **proper names** (e.g. “Süddeutsche Zeitung”), we used a workaround here to handle NOT FOUND nodes that contain proper names. In Section 5.4.2.2, we described this workaround, in which we locate proper names correctly in the profile graph by using an `object` value for them that is contained in the TGMOD.

The lack of coverage in GermaNet was especially noticeable in the area of **nominalizations** (“das Wandern” – i.e. “hiking” as a noun). For nominalizations, we tried to use contextual information (e.g. further words which are not nominalizations) given in the user answer for locating the information in the profile graph. Assume that the user has entered “Wandern gehen” (to go hiking), for example, and “Wandern” could not be found, but “Gehen” could be found in the GermaNet nouns. Then, the interest node with attributes <“object=Wandern, activity=gehen”> would be located under the GermaNet node “Gehen”. Semantically, this may not be the nicest solution, but it keeps the paths in the profile consistent (cf. Section 4.4.2.3) and helps to circumnavigate the restricted GermaNet coverage.

6.2.3.3 Target Group Matching and Target Group Model

From the 5 *milieus* originally suggested by [Schulze 1992] (cf. Section 5.3.1.2), we have implemented only four (excluding the Integrationsmilieu). The reason for this decision was that, with respect to the contained interests, the Integrationsmilieu is a mixture of two other milieus (Niveumilieu and Harmoniemilieu). For developing the TGMAT we wanted clear-cut TGs, however, in order to keep the decisions made by the TGMAT transparent and comprehensible. In addition, the error tracking in the development phase was easier for clearly distinct TGs.

Semi-Automatic Creation of the Target Group Model

As pointed out earlier, arbitrary TGMODs can be used within our profiling system that describe the interests of user groups by means of short natural language phrases comparable to our user input. The preparation of a TGMOD for system use is a semi-automatic process in which the designer of a specific application is supported by our system. The support can be achieved by means of a Target Group Design Tool (cf. Section 5.3.1.1). Although such a tool does not yet exist for LINGUINI, the basic functionality is already provided by our system components. In the following, we describe how we semi-automatically created our TGMOD by means of LA and PM.

In order to be able to analyze the natural language descriptions of interests contained in the model by [Schulze 1992] with our LA, we had to store the interests as plain text in the format expected by the LA (i.e. no parenthesis, no coordinations, semicolon used as interest separator, etc.). Furthermore, single-unit proper names (e.g. “Bildzeitung”), which are typically analyzed as `object` values by the LA, were slightly modified by us in order to be sure

that they can be found in GermaNet. The original interest “Bildzeitung” in Schulze’s model, for example, which would lead to a NOT FOUND node with `object = “Bildzeitung”`, was therefore transformed into “Zeitung, besonders Bildzeitung”. This led to the TG node `<object = “Zeitung”, favorite_items = “Bildzeitung”>` which could be found in GermaNet.

After this preparation, we could process the interests of the TGs with the LA and an interest node was created for each of them. We then used the PM to construct a graph from all interest nodes created for a TG, i.e. one graph has been constructed for each TG. After this, the TG graphs had to be adapted manually in the following respects:

- **Disambiguation:** For each interest we manually eliminated all paths found in GermaNet that represent different meanings, except for the one with the intended meaning.
- **Weights:** We added a `weight` attribute to each interest node in the TG (cf. Section 5.3.1.2).

Extension of the Original TGMOD with Further Interests

When testing our prototype, we found that the interests in the TGMOD based on [Schulze 1992] are sometimes not at all related to the interests entered by a user during a single dialog interaction. In this case, it is difficult for the DM to find meaningful further questions. We therefore extended the original TGMOD by [Schulze 1992] with about 40 additional, practically more often occurring interests, which we derived from the user data collected in our first study (cf. Section 6.1). Such interests, were, for example, “Freunde treffen (meeting friends)”, “Kaffee trinken mit Freunden (having coffee with friends)”, “Radfahren (cycling)”, “Bergsteigen (mountaineering)”, “Reisen (travel)”, “Wellness”, “Hunde (dogs)”, “Tiere (animals)”, “Spaziergänge (walks)”, etc. We added weights to the additional new interests on the basis of the frequency with which they occurred in our user study. By adding interests in the way described here, we were again able to significantly improve the performance of LINGUINI.

The Siblings Threshold Parameter

With respect to the **similarity relations** between nodes (cf. Section 5.3.2.4), we found that the TGMAT algorithm sometimes considered two nodes “siblings” although they were not. Therefore, we introduced a **configurable siblings threshold**. This threshold is set to a certain value (0.13 at the moment) which the similarity value of a node pair (cf. Section 5.3.2.4) has to pass in order for the node pair to get assigned the “siblings” relation. If one does not want to use this threshold, its value can simply be set to 0. With setting the value to 0.13, the “siblings” relation is assigned properly in the prototype system.

6.2.3.4 The Dialog Manager

The DM component has been implemented in its full complexity as described in Chapter 5.4 except for the following aspects. First, the prerequisites have been created for an easy realization of the concept of **attributes with a high application priority** (cf. Sections 4.2.2.2 and 5.4.2.4), but we have not yet implemented it. We have implemented the concept of nodes with high application priority, however, in order to show the general feasibility.

Secondly, a concept for a **word sense disambiguation approach** in LINGUINI has been sketched in Section 3.6.3 and the main functionality of the disambiguation component has been implemented within a diploma thesis [Eiseler 2005]. For technical reasons described in Section 6.2.3.2, we could not yet integrate this implementation in the LINGUINI prototype.

Finally, as we have described in Section 5.4.2.2, the DM performs an automatic relocation of NOT FOUND nodes, if possible. This functionality has already been implemented in the prototype. We have not yet implemented the possibility of **asking questions about NOT FOUND nodes**, a functionality which has been designated in our concept, however (cf. Section 5.4.2.5). The prototype could be extended to ask such questions with not too much effort because the PM is already able to update NOT FOUND nodes and to place them at the correct position in the graph.

6.2.3.5 The Question Generation

We already explained that we have not yet integrated a morphological processing into our Question Generation component, because this was not the main focus of our work (cf. Section 3.8.2). Although we have implemented a rather simple Question Generation instead, which does not generate word endings, the perceived quality and understandability of the questions proved sufficiently high and acceptable, as shown in our final user study (cf. Section 6.3.1.5 “3.2 Questions and answers”). By adding a morphological tool which creates correct word endings, the quality of the generated questions can be further increased.

6.2.3.6 The User Interface

For our final system evaluation, we have implemented a Web-accessible User Interface with typed input facilities as described in Section 5.4.1.2 (for screenshots of the User Interface cf. Section 6.3.1.3). It is based on the JAVA Servlet technology. We have not realized a speech interface for the prototype in this thesis, because our main aim was to provide the entire underlying functionality of the system with respect to dialog management and profile management as a first necessary step towards the direction of an adaptive profiling dialog. Since our system is also applicable with typed input for Internet and E-Commerce applications, suitable graphical user interfaces should be realized according to the requirements of different applications individually. For this purpose, our User Interface has been implemented in such a way that it can either be replaced easily by a more sophisticated interface or that it can serve as a basis for being adapted to the needs of specific Web-based applications.

6.3 Final Evaluation of Profiling Dialog and Profile Quality

We evaluated our prototype system by means of a user study in which subjects conduct a profiling dialog with LINGUINI. They were asked to assess the dialog afterwards by means of filling in a questionnaire. This study mainly aimed at evaluating the **adaptability of the dialog** and also the **overall acceptance** of the users. Another aspect of the study was to show the **development status** of the prototype and give suggestions for future improvements. Subsequently, we evaluated the objective **quality of the profiles** created in this user study. Here, the aim was to assess whether PM and LA have cooperated successfully in correctly processing and representing all information entered by the users. In the following, we first describe the user study which evaluated the profiling dialog (Section 6.3.1) and then the profile quality assessment (Section 6.3.2). We conclude with a summary and discussion of the results of both evaluation parts (Section 6.3.3).

6.3.1 A User Study about Dialog Adaptability, User Acceptance, and Development Status

6.3.1.1 Type and Aims of the User Study

The user study we conducted is a **usability test** of our prototype system. Usability tests are normally used to evaluate the interactive properties of software systems. For the study, we decided to use the technique of an **interview in written form** (cf. [Bortz and Döring 2002]), in which the subject writes down answers to a set of definite questions in a questionnaire, and not the **oral interviewing technique** in which the user freely narrates with only a few guiding questions given and the interviewer takes notes. The first method seemed more adequate for the evaluation of a software prototype and the individual answers of the subjects (to the mainly quantitative questions) can be summarized and compared much better than with the second technique. Furthermore, the answers given by subjects are often less biased in an anonymous questionnaire than they are if an interviewer is present.

A special problem we had to solve was the following. Since the dialog adapts to each user individually, the order and type of questions asked is different for each user. For this reason, we could not directly compare the course of the different dialogs with each other and determine their quality on this basis. Instead, we had to find specific evaluation criteria that apply to all dialogs (even though the course of the dialog is different for each user) and that can be compared. We decided to evaluate **three main criteria**, for which we formulated specific questions. These questions are stated in detail together with the results in Section 6.3.1.5. Here, we only introduce the three criteria briefly:

- An important aspect that applies to all types of dialogs is the degree of **user acceptance**. How did users feel about the dialog with LINGUINI? To what extent are they willing to use it in the car scenario or in other application scenarios?
- Since the adaptive capabilities of LINGUINI are the main reason for this lack of comparability, **dialog adaptability** should also be an interesting criterion to evaluate. Besides this, adaptability is a central feature of our dialog approach. Here, we want to know to what extent the subjects perceived an adaptability of the profiling dialog and how important adaptability is for them in such a dialog. Did they like the new topics suggested by the system and did they perceive them as related to their interests?
- A third criterion is the **development status** of the prototype as perceived by the users. Did users notice any errors, how irritating were these errors? How did users find the formulation of the questions and the different answer possibilities? What kind of improvements and extensions would be desirable in their opinion for using LINGUINI in practice?

What we did not want to evaluate with this study was the usability of the Web User Interface, because LINGUINI is intended to be used finally with a speech interface. A second reason was that the User Interface we used for the study is quite simple and is not comparable to fully elaborated user interfaces of E-Commerce platforms, which can be expected to perform better with respect to usability. Above all, the functionality of the adaptive dialog system was the main aspect of this thesis and should hence be evaluated in the first place here.

6.3.1.2 The Questionnaire

For the study, we have designed a questionnaire, which is shown completely in Appendix B.1 of this thesis. Helpful suggestions for the design of the questionnaire were taken mainly from

[Bortz and Döring 2002]. The questionnaire contains **35 questions** about the conducted dialog and some demographic questions about the subjects. Of the 35 questions, **31** were **quantitative** (subjects chose from a defined metric scale or numeric values) and **4** were **qualitative** questions (subjects stated their opinion or experience freely).

The questions have been bundled into six clusters that contribute to the investigated criteria as follows. Note that we have assigned the clusters to one of the three main criteria to which they *mainly* contribute (occasionally an individual question may also contribute to another criterion):

1 User acceptance

- Anwendung (application)
- Interesse und Empfindung (interest and emotions)
- Dauer des Dialogs (dialog duration)

2 Dialog adaptability

- Anpassungsfähigkeit des Systems (adaptability of the system)

3 Development status

- Vorschläge zur Verbesserung (suggestions for improvement)
- Systemvorgaben und Fragestil (answer options and questioning style)

Oftentimes, one cluster contains several questions that are *variants of the same statement*, i.e. their formulation differs only slightly. This is common practice in user studies in order to further substantiate an answer given for one question by means of answers given for other questions that refer to the same statement. It also helps to avoid that subject and questionnaire designer interpret the same statement differently.

After the questions of the six clusters, the last part of the questionnaire asks about personal data of the subjects, such as age, gender, experience with computers and dialog systems, education and current profession.

6.3.1.3 Preparation and Test Procedure

We set up a Web server on which LINGUINI was running and configured the prototype with optimal parameter settings (see below). The subjects were invited by email to participate in the study. With this email, the subjects received the instructions and user guidelines (see below and cf. Appendix B.2) as well as the questionnaire they had to fill in (cf. Section 6.3.1.2 and Appendix B.1). The subjects could access LINGUINI via a Web interface and conduct a profiling dialog, i.e. enter their interests in natural language by means of typed input and output. Users were allowed to end the dialog at anytime, even before the official end of the dialog. For each subject, a user profile has been created and stored in the database. After the dialog interaction, the users were asked to fill in the questionnaire, which they could send back by email or fax. After all questionnaires were received, we evaluated them anonymously. The quality of the created profiles was evaluated after the user study (cf. Section 6.3.2). Some sample dialogs conducted in this user study can be found in Appendix B.3.

In the remainder of this section, important aspects of the test preparation and test procedure are discussed in more detail.

Parameter Settings

The system parameters “dialog length” and “nodes with high application priority” (cf. Section 4.2.2.2) have been set as follows. The **parameter “dialog length”** was set to a maximum of

30 questions, but shorter dialogs were possible for individual users, because LINGUINI adapts the length automatically to the course of dialog (cf. Section 5.4). The number of 30 questions was chosen on the basis of tests we had conducted previously (cf. Section 6.2.2.3). One aim of the study was also to find out whether this was a realistic dialog length accepted by the users. The **parameter “nodes with high application priority”** was instantiated only with the node “Theater (theatre)” in order to show that its functionality works in general. Hence, the system asked about the topic “theatre” first, in case this was detected as an interest of the user. We did not want to instantiate the parameter with further nodes with high application priority, because this would influence the topics of the dialog too strongly in this general study.

The other configurable parameters (described in Section 4.4.2.2 and Chapter 5) were set to their optimal values, which we also determined empirically by a large number of previously conducted tests.

User Guidelines and Instructions for the User Study

Before the user study, the subjects were given user guidelines and instructions as to how to perform the test. A large cutout of this document can be found in Appendix B.2. Users were asked to read the document thoroughly before the test and to stick to the guidelines as closely as possible. At the present development status of the prototype, disregarding the guidelines may cause major problems during the dialog and may hence lead to poor judgments in the user questionnaire.

In this document, we also explained briefly the purpose of the study and the tasks and characteristics of LINGUINI. The subjects were informed that the prototype does not yet have a morphological question generation (cf. Section 3.8.2) and that some questions therefore may sound slightly ungrammatical. They were also informed about the type of user input expected. This includes the avoiding of typing errors, because we have not integrated a spell checker into the prototype, or colloquialisms, because these are difficult for NLP systems in general. Moreover, they were asked to enter nouns with a capital first letter, because otherwise this might cause problems for the part-of-speech tagger. Examples were given which illustrated how interests should be formulated such that LINGUINI is able to understand them.

Before conducting the dialog, the subjects were asked to imagine the in-vehicle scenario sketched in Chapter 1, i.e. a longer highway trip on which they conduct the dialog for getting event and sightseeing recommendations for their travel goal. We wanted the subjects to imagine this situation intensely beforehand because several of the questions in the questionnaire referred directly to this application situation.

The subjects were also told that LINGUINI would terminate the dialog if it had obtained enough information, but that they could end the dialog at any time beforehand by entering “ende (end)”.

The LINGUINI Web Interface

The subjects were able to access LINGUINI, which was running on a server, via the Internet. They first had to log into the system, then they reached the LINGUINI test user interface shown in Figure 6.2.¹⁷ The starting question (cf. Section 4.2.1.1) was shown:

¹⁷ We modified the original SFB 582 platform, such that users were only shown the profiling dialog system and no other functionalities.

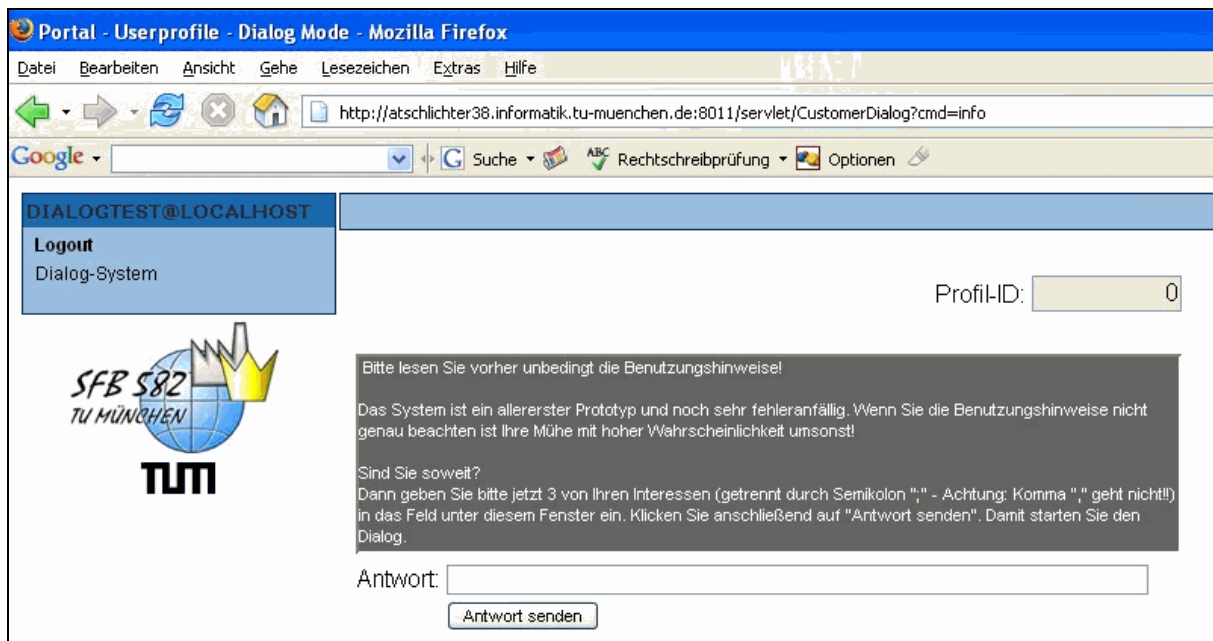


Figure 6.2: LINGUINI test interface with starting question

The subjects could enter their answers in the field below the system question and then send their answer by pressing the button. As a consequence, the next question occurred in the question field above (cf. Figure 6.3). In the grey field at the bottom, the subjects could scroll through the dialog course thus far:

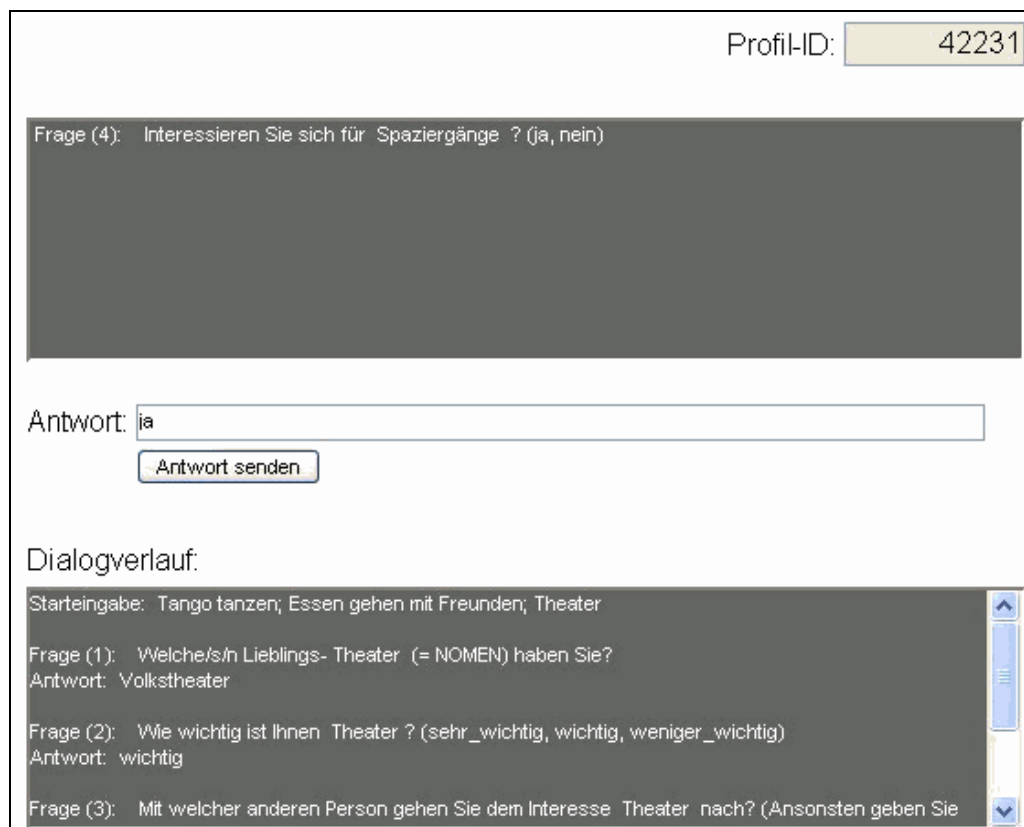


Figure 6.3: Question and answer windows (top) and course of the dialog (bottom)

During the course of the dialog, a profile was created for the subject and its profile ID was shown on the top right of the window (cf. Figure 6.3). The users were asked to enter this profile ID into their questionnaires, such that we could evaluate the profile quality for this dialog course afterwards.

6.3.1.4 The Participants of the User Study

Altogether, 30 users took part in the study. From the set of 30 questionnaires returned by the users, only one questionnaire had to be discarded because the majority of its answers are “outlier values” (or: “outliers” [Bortz and Döring 2002]) as compared to the answers of all other participants. Statistically, outliers are not representative for the rest of the participants and therefore have to be removed from the data set in order not to adulterate the overall results (cf. [Bortz and Döring 2002]). In Appendix B.4, we illustrate the values of this questionnaire and explain in more detail why it could not be used for the evaluation.

From the **29 users** providing valid questionnaires, 14 are **female**, 15 **male**. Their **ages** range from 20 to 55 (35.6 in average). The subjects did not know the implementation or technical details of the test prototype. They have different **professional and educational backgrounds**: 9 have IT-related professions or educations, 13 have neither IT-related professions nor IT-related educations, and for 7 this is unknown (the total number of subjects is indicated in parenthesis):

- **IT-related (9):**
Computer scientists (4), IT consultants (2), developer (1), management consultant/computer scientist (1), IT manager (1)
- **Not IT-related (13):**
Business men/women (3), engineers (2), psychologists (2), communication and media scientist (1), author (1), school director (1), translator (1), lawyer (1), artist (1)
- **Unknown whether IT-Related (7):**
Research employees (3), students (3), PhD student (1)

All subjects have used dialog systems before (e.g. speech dialog systems at telephone hot-lines, virtual assistants on the internet, etc.). **19 subjects** even indicated that they had **frequently used dialog systems** before.

The subjects judged their **general computer skills** as follows:

- 22 rather or very skilled
- 6 middle-rate skilled
- 1 less skilled

6.3.1.5 Results

The three main topics we wanted to investigate with the user study were user acceptance, dialog adaptability, and the development status of our prototype. For each of these topics, we formulated specific questions, which will be presented here together with the results from the user study. All results will be discussed and interpreted in Section 6.3.3.

In case a question in the questionnaire has been answered by fewer than all participants, the number of received answers is indicated in parenthesis (e.g. N=28). “N=28” means, for example, that 28 users (from 29 users overall) have answered this question.

1 User Acceptance

For this aspect, we also investigated whether there are correlations between the **computer skills** or the **gender** of the users and their **willingness to use LINGUINI**. No such correlations could be found, however, i.e. computer skills and gender do not seem to influence the user's willingness to use LINGUINI in general.

1.1 Emotions:

How were the user's interest and feelings with respect to the interaction?
What did users like about the interaction with LINGUINI?

The **interest** of users in LINGUINI is represented by Q30¹⁸ in the questionnaire. Figure 6.4 shows that most users found it quite or completely interesting to talk to LINGUINI:

- 66 % of the users found it **quite or completely** interesting
- 17 % found it **middle-rate** interesting
- 17 % found it **scarcely or not** interesting

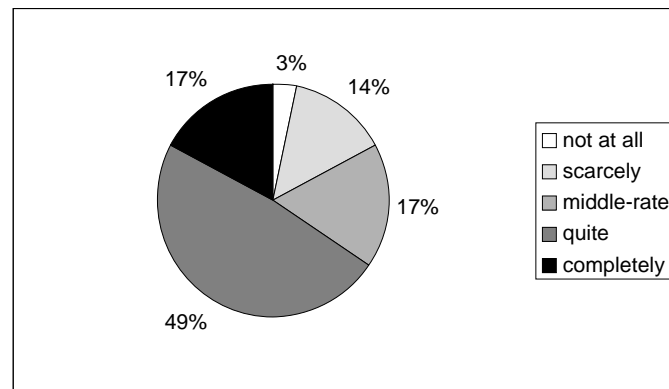


Figure 6.4: Q30. Did you find it interesting to talk to the system?¹⁹

The **feelings** of users during the interaction are indicated by Q31. First, Figure 6.5 illustrates the rather **positive emotions** of fun and excitement. The overall result is that most users clearly indicated positive emotions while using the system:

- 19 users found it quite or completely **funny** to talk with LINGUINI
8 users found it middle-rate **funny**
1 user found it scarcely **funny**
- 15 users found it quite or completely **exciting** to talk with LINGUINI
10 users found it middle-rate **exciting**
4 users found it scarcely **exciting**

¹⁸ Q30 means „Question No. 30“ in the questionnaire (cf. Appendix B.1).

¹⁹ In the figures, we indicate a shortened form of the questions in English. The original questions formulated in German can be found in the questionnaire in Appendix B.1.

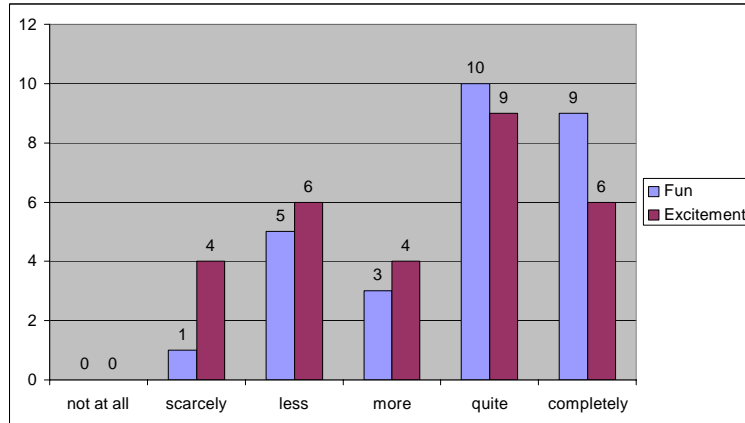


Figure 6.5: Q31. Which sentences match best your feelings during the dialog and how much:
 A1: The dialog was fun (0 = not at all, ..., 5 = completely).
 A2: The dialog was exciting (0 = not at all, ..., 5 = completely).

Secondly, Figure 6.6 illustrates the **rather negative emotions** of boredom and annoyance:

- 19 users felt scarcely or not at all **bored**
 7 users felt middle-rate **bored**
 2 users felt quite **bored** while talking to LINGUINI
- 19 users felt scarcely or not all **annoyed**
 8 users felt middle-rate **annoyed**
 1 user felt quite **annoyed** while talking to LINGUINI

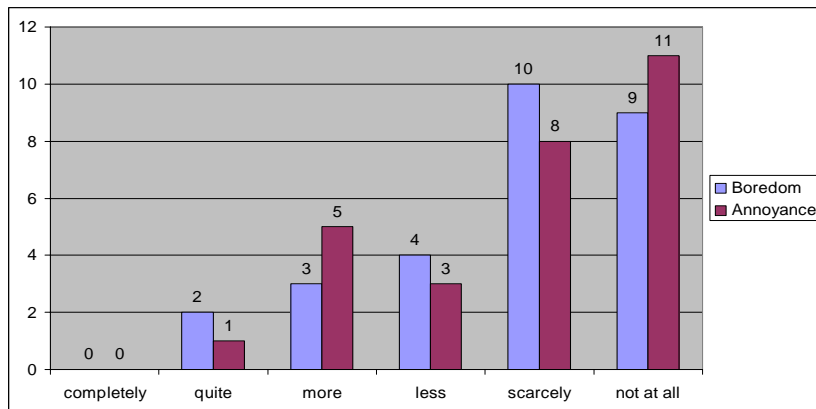


Figure 6.6: Q31. Which sentences match best your feelings during the dialog and how much:
 A3: The dialog was boring (0 = not at all, ..., 5 = completely).
 A4: The dialog was annoying (0 = not at all, ..., 5 = completely).

Q32 was a free text question in which we wanted to find out what users liked about LINGUINI. We got the following feedback from 22 users (some stated several of these items):

- 8 users mentioned that they particularly liked the fact that the system asked them about interests which were related to their own and that the system was capable of making its own suggestions adaptively.
- 5 users stated that they especially liked the adaptation of the system to their answers and the flexibility of the dialog to be different for each user.

- 4 users particularly liked the feeling of being taken seriously and that the system seemed to “give its time and attention” to them.
- 3 users indicated that they especially liked the fact that they could enter free text answers.
- 3 users liked the clarity and understandability of the questions
- 2 users found the questions about the attribute `location` particularly inspiring
- 2 users liked most of all that the dialog was amusing and interesting

1.2 Adequacy of dialog length:

How did users perceive the length of the dialog in the user study?

Although the users were allowed to terminate the dialog at any stage, none of the subjects made use of this option. All users conducted the dialog until the system terminated it.

Q22 and Q23 are variants of the same question statement: Was the **dialog length** perceived as ideal, too long, or too short **for getting to know the user sufficiently** for the task of generating personalized recommendations? Figure 6.7 illustrates Q22 and shows that **48 % of the subjects found that the dialog length was ideal** for getting to know their interests. 38 % found that a longer dialog would have been necessary in order to get to know them sufficiently. 14 % found that it should be shorter for this task.

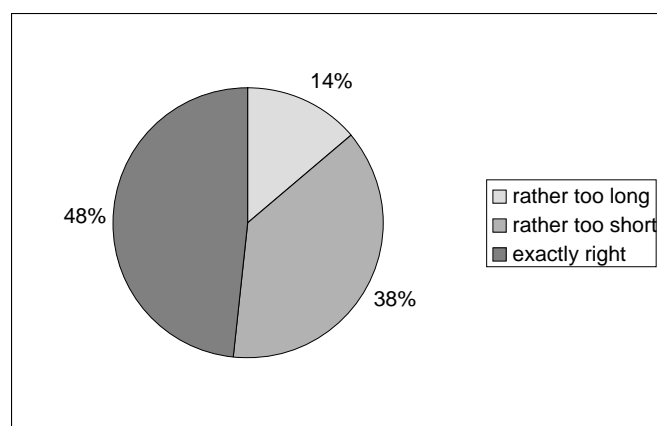


Figure 6.7: Q22. The system has the task to get a very exact picture of your interests in order to generate adequate recommendations (e.g. about events and leisure activities) for you.

Did you find the duration of the dialog rather too short or too long for this task?

This result is substantiated by Q23, in which about **two third of the users found that the dialog took long enough** to get a sufficiently exact picture of their interests, whereas only about one third found that this was not the case.

1.3 Application:

Are users more willing to reveal information if they can do this in parallel to some other task?

Would they like to use LINGUINI especially in the vehicle scenario?

Do users prefer speech over text interaction in general, even at home at their computer?

Which other situations would be attractive for conducting such a dialog?

The **willingness of users to reveal information if they can do something in parallel** is represented by Q26 (cf. Figure 6.8). The overall result is that for most users the motivation to reveal information is higher if they can do something in parallel. For 28 % the motivation is extraordinarily higher and for 41 % it is quite higher. For 14 %, the situation does not seem to make a difference. For 17 % the motivation is scarcely higher if they can do something in parallel, and no users indicated that their motivation would be not at all higher then.

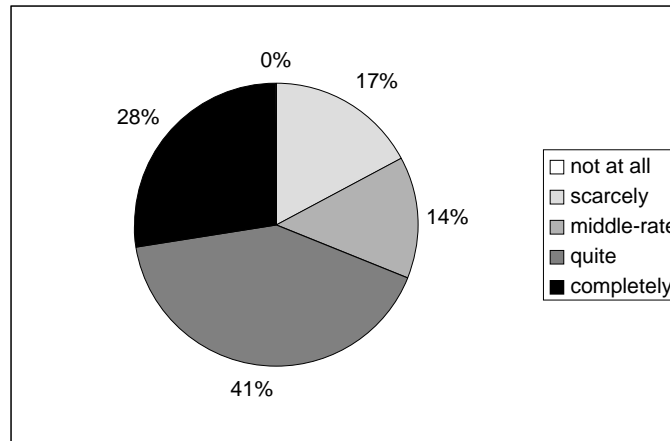


Figure 6.8: Q26. The dialog can also be conducted via speech interface while pursuing some other activity (e.g. driving, walking, domestic work, etc.). Would you prefer to tell the system your interests while doing something else in parallel rather than sitting at the computer?

The **willingness to use LINGUINI especially in the vehicle scenario** is represented by Q25 (cf. Figure 6.9). The overall result is that **three quarters of the users would like to use LINGUINI especially in the vehicle scenario** for getting sightseeing and event recommendations. The details are as follows: 39 % would very much like to use it in this scenario, 36 % would quite like to use it there, and 18 % would middle-rate like to use it. Only 7 % would rather not like to conduct a dialog with LINGUINI in this scenario.

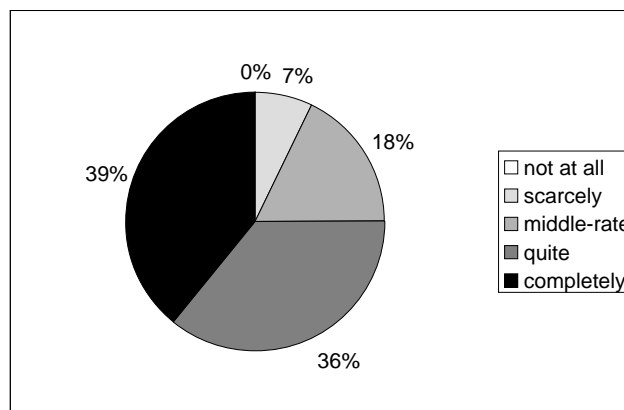


Figure 6.9: Q25. How much would you like to use this dialog system (with spoken language) in the just described vehicle scenario?

Q27 reveals that for 23 users the motivation to conduct this dialog was **higher on a longer highway trip than if they had to do it at home at their computers**. 4 users indicated that their motivation was lower on a longer highway trip and for 2 users this did not make a difference.

With Q29, we wanted to investigate whether the speech capability of LINGUINI can be considered an important step towards developing explicit profiling methods, because it may lead to a higher user acceptance in general, not only in speech-requiring situations. The answers given in Q29 show that 7 users **would prefer a speech interaction even at home at their computers**, while 22 users prefer typed input and output for a profiling dialog at their computers. Hence, the need for speech profiling is clearly given for a quarter of the subjects.

Q28 investigates whether the users can imagine other everyday situations which would be suitable, in their opinion, for conducting such a dialog (either via keyboard or speech interface). In the following, we give some sample ideas users mentioned about situations – other than the vehicle scenario – in which they would particularly like to use LINGUINI and also about possible applications for which they could imagine to use it:

Suitable situations for acquiring information with LINGUINI (via keyboard or speech):

- **Sports:** fitness studio, jogging, etc.
- **Waiting periods & travel:** at the doctor's, at authorities, when waiting for public transportation, in trains, on air trips, etc.
- **Relaxing:** in a comfortable armchair, sofa, hammock, on the balcony
- **Manual of creative work:** domestic work (cooking, cleaning, etc.), gardening, baby sitting, painting, doing handicrafts

Possible applications for information acquired with LINGUINI:

- **Online shopping:** for getting better product recommendations; selecting Xmas presents; music, movie, and book recommendations
- **Offline shopping:** system recommends shops in a particular city; user is looking for something specific and system tells her where she can buy it in this city; recommendation of items located close to the actual position of the user via mobile device
- **Event and leisure planning for home town:** event suggestions for home town; spontaneous checking as to what evening events are possible (e.g. one has missed a movie and quickly wants to find out which other movies are shown in town that start in the next hour); leisure recommendations for new citizens of a town
- **Travel planning:** recommendations for tourists about events and sights in a city; route planning; “digital information column” at a tourist location for more personal information than is generally possible; finding the right holiday options, destinations, and activities in a travel agency or online
- **Dating agencies**
- **Job search:** finding a suitable job at the job center (user profile could be compared with “job profiles” and “company profiles”)
- **Diagnosis systems:** in the medical area and for finding computer errors
- **Further desired types of recommendations:** of Internet sites; of cooking recipes

2 Dialog Adaptability

2.1 Importance of adaptability:

How important is dialog adaptability for the user?

Figure 6.10 shows for Q11 that **adaptability and personalization is in fact highly important** in a profiling dialog for 90 % of the users. In detail, 41 % consider it completely impor-

tant that the system adapts to their personal interests, 49 % consider this quite important, and 10 % feel indifferent. No users consider the adaptation to their personal interests scarcely or not at all important.

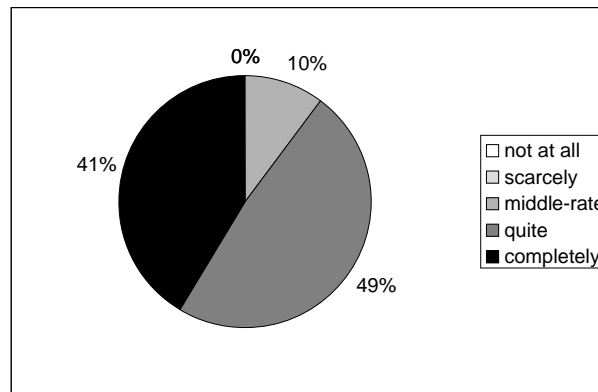


Figure 6.10: Q11. How important is it if for you that the system adapts to your personal interests in such a dialog?

This result is substantiated by Q13, which asks, whether users would rather conduct a dialog with a system that does not adapt to their personal interests and asks standard questions in a fixed order that are the same for each user. Only 2 users (7 %) indicated that they prefer a non-adaptive, standardized dialog, whereas **the majority of 27 users (93 %) indicated that they would prefer the adaptive dialog.**

2.2 Adaptation to mentioned interests:

To what extent did the system adapt to and dwell on the interests mentioned by the user?
How satisfied were the users with the level of detail reached during the dialog?

Figure 6.11 shows, **how much** users perceived the dialog to **adapt to and dwell on their personal interests** (Q1) and how **satisfied** they were with this degree of adaptability (Q12). The results show that most users perceived the system as middle-rate adaptive and were middle-rate satisfied. Furthermore, about the same number of users were quite or completely satisfied as were scarcely satisfied.

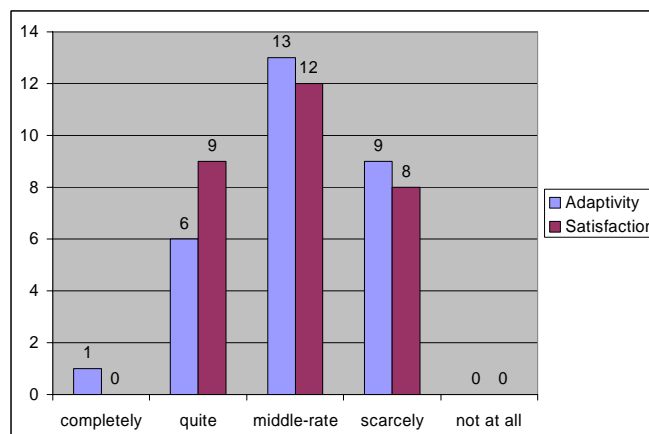


Figure 6.11: Q1. How well did the dialog adapt to and dwell on the interests you mentioned?
Q12. How satisfied are you with the degree to which the system dwells on you personally?

Q7 investigated whether users would have **liked to talk about some of their interests longer**. Here, 20 users (~ 70 %) answered “yes”, whereas 9 users (~ 30 %) answered “no”. The results from Q8 shown in Figure 6.12 indicate that **55 % of the users felt that the system had not asked enough details** about the individual interests. A quarter felt that the system has asked too many details and about the same number found the number of details exactly right.

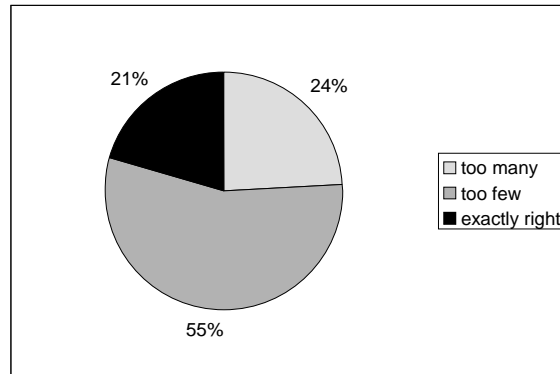
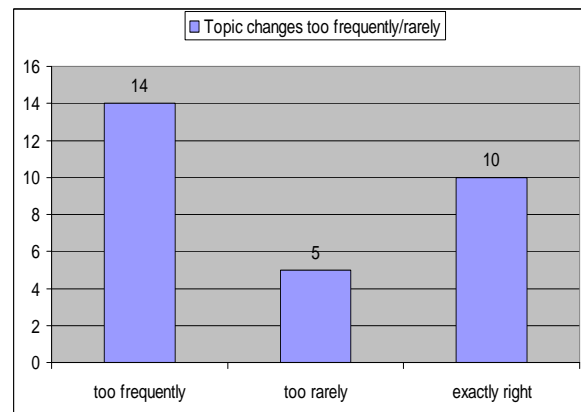
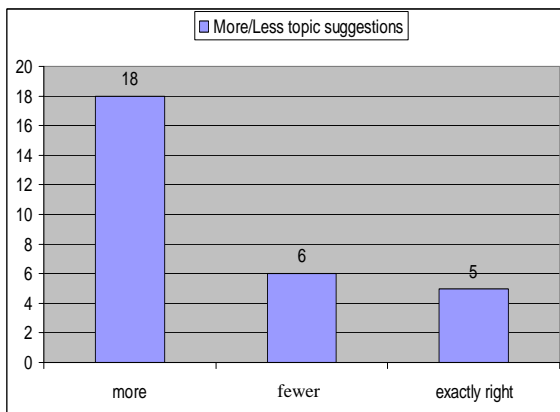


Figure 6.12: Q8. Did you have the impression that too many or too few questions about details of specific interests have been asked in general?

Hence, it seems most users would have liked to talk longer about individual interests by being asked more details about them. This result is substantiated by the following observations presented in Figure 6.13. Q9 showed that half of all users wanted the system to **make fewer suggestions for new topic**. The results for Q10 show that about two third of the users think that **the system changes the topic too frequently**.



Q9. Should the system suggest more or fewer further/new topics?

Q10. Did you find that the topic is changed too frequently or too rarely during the dialog?

Figure 6.13

2.3 Suggestion of related topics:

Were new topics suggested by the system semantically related to the interests mentioned by the user?

Were the topics suggested by the system perceived as reasonable?

The first aspect investigated here is the thematic relation between user interests and new topics suggested by the system. Figure 6.14 shows whether users found that new topics suggested by the system were **thematically related to their interests** (Q4). For about half of the users this was frequently or very frequently the case, for about one third this was occasionally the case. Figure 6.14 also shows that for the majority of users (76 %), the newly suggested topics showed an **understandable relation to their interests**, which was especially expressed by questions of the type “Da Sie sich für X interessieren, interessieren Sie sich auch für Y (Since you are interested in X, are you also interested in Y)?” (Q5).

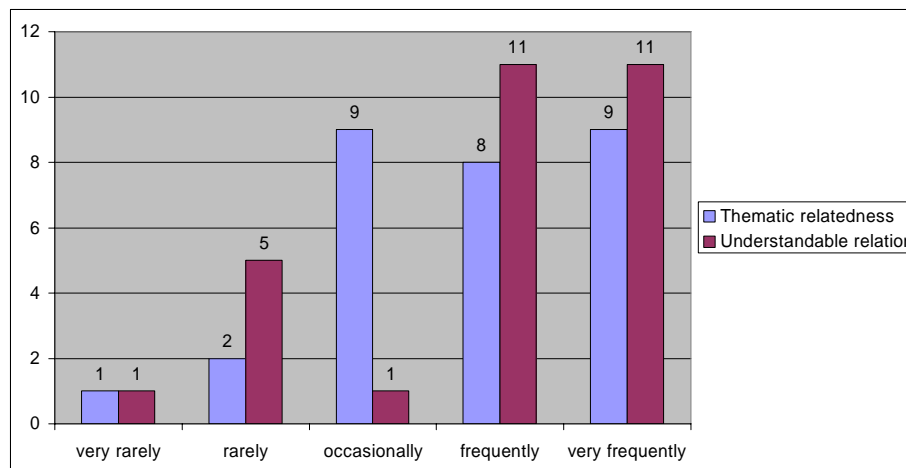


Figure 6.14: Q4. Were new topics suggested by the system thematically related or similar to your own? Q5. Sometimes the system relates one of your interests to a new topic (e.g. “Since you are interested in X, are you also interested in Y”). Did you find the correlation between your interest and the new topic understandable in general?

The second aspect we investigated was whether users found the topics suggested by the system reasonable. Q6 (N=28) indicates the reasonability of the questions:

- 36 % found the suggested topics **very frequently** reasonable
- 32 % found the suggested topics **frequently** reasonable
- 21 % found the suggested topics **occasionally** reasonable
- 11 % found the suggested topics **rarely** reasonable
- 0 % found the suggested topics **very rarely** reasonable

Finally, we also investigated whether users found new topics suggested by the system interesting in general (Q2) and how inspiring users found new topics suggested by the system (Q3). The answers to these questions did not show a homogeneous picture, however, that allows for a clear interpretation (cf. Section 6.3.3).

3 Development Status

3.1 Perceived errors:

Did users have the impression that the system made errors?
How irritating were these errors?

Q33 investigated whether the users had the impression that the system makes mistakes. The overall result is that more than 50 % of the users found that the system made mistakes rarely

or very rarely, whereas 45 % found that this occurred occasionally. No users indicated that errors occurred frequently or very frequently.

Q34 (N=28) elicited whether these errors were irritating for the users. The overall result is that for 61 % of the users the errors were scarcely or not at all irritating, 18 % of the users found them middle-rate irritating, and 21 % found them quite irritating.

3.2 Questions and answers:

Did the users understand the questions and like their style?

Were free answers and answer choices well-balanced in the dialog?

Figure 6.15 represents Q14 and Q15 and shows the opinion of the users about the **style of the questions**. In Q14 users were asked to judge the formulation of questions, apart from the fact that word endings are sometimes incorrect, because no morphological component has been integrated in the Question Generation. The answers for Q14 show that the formulation of the questions was considered quite or completely good by 23 users.

With Q15, we investigated whether users understood what the system meant with a question. Figure 6.15 shows that the questions could be understood well by the users in general.

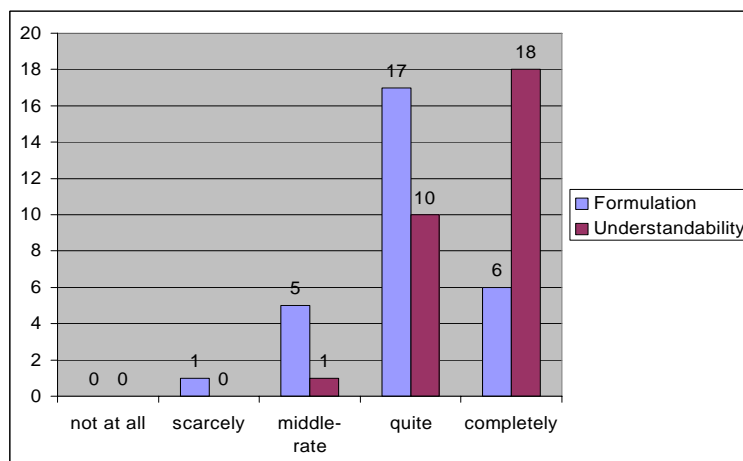


Figure 6.15: Q14. Did you find the formulation of the questions appealing (e.g. was the length of the question ok, did the question style sound natural, etc.)?

Q15. Did you understand the contents of the questions?

With Q20 and Q21, we investigated whether users found the mixture of free answers and answer choices well-balanced in LINGUINI.

- Q20: 66 % of the users found the number of **free answers** exactly right in LINGUINI, 24 % wanted freer answers, and only 10 % wanted fewer free answers.
- Q21: 69 % of the users found the number of **predefined answers choice questions** exactly right in LINGUINI, 17 % wanted more predefined answer choices, and 14 % wanted fewer of them.

Finally, Q16 and Q17 investigated whether users found the fact that the system suggested new topics itself relieving (Q16) and whether they would have liked to get more suggestions from the system (Q17). Q18 and Q19 investigated whether users found it difficult to express their interests as free answers (Q18) or whether they found it difficult to express their interests by selecting predetermined answer values (Q19). The answers to these questions did not show

a homogenous picture and hence did not allow for a clear interpretation, which is the reason why they are not described in more detail here.

3.3 Improvements:

Did users have any suggestions for improving the dialog, and which additional functions would they like to have?

With the free text question Q35, we intended to collect ideas for future improvements of the system and about functions users consider desirable. We also wanted to see whether some of the aspects we have elaborated on in our concept, but which have not yet been implemented in the prototype, are in fact perceived as missing or important. Here is a short summary of the most interesting comments:

Semantic checks as to whether attribute questions make sense:

Questions about `locations` or `participants` are sometimes not semantically appropriate (e.g. `participants` for “reading” or `locations` for “movies”).

More possibilities for the user to control the dialog flow:

Users suggested enabling them to select topics and to order topic blocks that are addressed during the dialog according to personal relevance and taste.

Improvement of fallback questions:

Instead of asking very general fallback questions at a rather late state during the dialog, the system could provide free associations about new topics randomly (i.e. not adaptively) in order to help users remember and find more interests. Fallback questions could also be asked more specifically, e.g. about interests only relevant at the weekend, in the evening, etc.

Insight into and feedback about the profile data stored:

Users wanted to get some feedback about what the system has learned from their input and wanted to be able to check whether the system has understood them correctly. They also wanted to see their profile after the dialog. Users also suggested that feedback should be given, if the system accomplishes a disambiguation of user input automatically, in order to find out whether the system has chosen the correct interpretation.

Other improvements:

Other improvements were suggested for the design of the User Interface (which was only a simple test user interface), for example, to offer dropdown menus or radio buttons for the pre-defined answer choices. Users also desired a higher fault tolerance for the user input.

The results presented above will be interpreted and discussed in Section 6.3.3, together with the results of the profile assessment, which we describe in the following section.

6.3.2 Profile Quality and Coverage Assessment

6.3.2.1 Aims of the Quality and Coverage Assessment

The intention of the profile quality and profile coverage evaluation was to complement the user study described in Section 6.3.1 with an objective assessment of the profiles created during the dialogs. Hence, the profile assessment was conducted *after* the user study and without the help of the subjects. Each dialog conducted during the user study was stored in a separate log file together with the ID of the profile that has been created during this dialog. This way we could compare the dialog course and the profile after the user study. The major aim of this

comparison was to find out how well the profile management worked during the dialogs and whether problems caused by the profile management were noticeable for the users or correlated with the number of questions asked.

We investigated the profile created for each user during the user study by means of specific criteria. First, we wanted to find out the **number and types of errors** that occurred. We classify them according to the following categories:

- Errors caused by the external components GermaNet and TreeTagger
- Errors caused by the LINGUINI components we have developed. Here we distinguish
 - **Minor known errors**, i.e. errors we knew about before the user study, but the fixing of which would not have led to noticeable improvements.
 - **New errors or design problems**, i.e. errors we did not know of before and that were brought to light by means of the user study as well as aspects that have led to problems in the dialog, which we have not yet considered in the design.

Furthermore, we distinguished cases in which **additional functionality** was missing and led to problems during the dialog. With “additional functionality” we mean all types of functionality which we had designed and included in the concept already, but which we decided not to implement in favor of more important functionality.

Finally, we wanted to find out whether there are **correlations** between the number of questions asked in a dialog, the overall quality of the questionnaire (i.e. the subjective judgment of the user about this dialog), and the total number of errors that occurred for the dialog with respect to profile quality and profile coverage.

6.3.2.2 Results

Errors caused by GermaNet and TreeTagger

For the 29 dialogs conducted in the user study, 79 interests could not be found in **GermaNet**. These are 15 % of all 524 interests entered during the 29 dialogs. Not finding them in GermaNet caused interests to be stored under the NOT FOUND node and excluded them from further questioning. This means 2.7 interests in average per profile could not be found in GermaNet (from 18 interests entered in average per profile). It occurred three times that 6 interests (from 18 interests in average) could not be found in GermaNet within a single dialog and once 8 interests could not be found within a single dialog.

As opposed to this, there were only a few misinterpretations (i.e. wrongly assigned part of speech tags because of missing contextual information) caused by the **TreeTagger** that had an impact on the profile quality or coverage. These misinterpretations are described in more detail below.

Errors caused by Other LINGUINI Components and Desired Additional Functionality

Interestingly, almost all problems discussed here are related to missing, wrong, or overwritten **attribute values**. Only a few are related to missing or double **profile nodes**. This might be due to the fact that we could not improve and extend the attribute handling as much as desired in the time given. Many problems only afford minor changes, however, that can be achieved with not too much effort.

First, we noticed the occurrence of the following types of **minor known errors**. In average, 3.8 errors of the type “minor known errors” occurred for each profile:

- If the user enters **two or more information items** (separated with semicolons in the user answer) as answer to a deepening question, these are currently also **stored with semicolons** in the profile (e.g. `location = “München; Paris”`). This is not correct, because the only legal separator of multiple attribute values in the profile (not in the user answer!) is the comma. Hence, two attribute values in the profile that are separated by any symbol other than comma are interpreted as one value. The correct storage format would be, for example, `location = “München, Paris”`, in which the semicolon in the user answer has been converted into a comma in the profile.
- If an attribute contains multiple values, these **values sometimes occur several times** for this attribute (e.g. `activity = “lesen, schreiben, lesen, schreiben (reading, writing, reading, writing)”`).
- A user interest which cannot be found in GermaNet is normally stored as child of the NOT FOUND node. If the user enters the same interest once more, the **same interest is erroneously stored a second time** as child of the NOT FOUND node.
- Occasionally, a **new user interest, attribute, or attribute value** has not been stored in the profile at all. Although this error seems rather severe, it occurred very rarely. The cases in which it occurred do not seem to have a common cause. For correction of the error, each case would therefore have to be investigated and treated separately.

Secondly, we found two types of **new errors/design problems**, which we had not known or considered in the concept before:

- **An attribute value** set in a specific profile node (e.g. `location = “ins Kino (to the cinema)”` in the node “ins Kino gehen (going to the cinema)”) **is overwritten** later in the dialog with some other value (e.g. with “in die Oper (to the opera)” from “in die Oper gehen (going to the opera)”). The reason is that the PM recognizes that the name giving part of the profile node is the same in both cases (e.g. `activity = “gehen”`) and does not take into account the other attributes in the nodes. It selects the node it has created for “ins Kino gehen” again later on when the user talks about “in die Oper gehen” and overwrites the `location` value.
- The LA is already able to compensate some cases in which the TreeTagger **wrongly assigned part-of-speech tags** because of missing contextual information in the user input. Other tagging errors only occurred during the user study for the first time and the LA could therefore not yet handle them. They hence led to wrong attribute values. An example is “Berge (intended meaning: mountains)”, which is ambiguous for the tagger and has been interpreted by it as a verb form (derived from “bergen (rescue)”) and has hence been stored as `activity` and not as `object` value. Strictly speaking, this is a problem of the external component TreeTagger, but since many of these misinterpretations can be circumnavigated by extending the LA, we list this problem here.

These new errors/design problems occurred not as frequently as the minor known errors. In average, 0.5 of them occurred per profile.

Finally, we found three different types of **additional functionality** that would have been desirable for further improving the dialog. All of them have already been considered in our concept, but have not been implemented for the prototype. They mainly affect the extraction or correct assignment of attribute values for different reasons:

- Parts of the user input **cannot be handled by the LA** as yet. An example is “sich unterhalten lassen (to be entertained)”, in which the entire expression should be treated as a value for `activity`. At the moment such complex verb phrases cannot yet be handled. Another example is determiners (“the”, “a”, etc.) or pronouns (e.g. “viele (many)” in “viele Fotos machen (taking many pictures)”), which are not yet stored as part of an attribute value.
- The lack of a **disambiguation component** sometimes causes attribute values to be assigned wrongly. An example is the interest “Essen gehen (to eat out)” in which “Essen” has been assigned as `object` value correctly, but also as value for `location` (because it is contained in the Semantic Classes Lexicon as a German town/location). The disambiguation would help to decide which meaning of “Essen” the user has meant and would avoid the storing of additional meanings in the same interest node.
- Attribute values are assigned wrongly because the **Semantic Classes Lexicon** (which has been derived from GermaNet) contains words that are not representative for a semantic class or lead to problems with more frequently occurring words that have a different meaning. An example is “in ein Café gehen”, where “in ein Café” has been assigned as a `frequency` value wrongly, because “in” is stored as an expression related to time (“something is in/trendy”) in the Semantic Classes Lexicon. Hence, for the word “in”, the less frequent meaning of “being trendy” has been confused with the more frequent meaning of “being inside”.

The effect of missing additional functionality was not very grave, however. It led to only 0.6 errors per profile in average or in other words: 0.6 errors could be avoided in average per profile in the future, if these additional functionalities were implemented.

Correlations between Dialog Quality and Profile Quality

Besides investigating criteria for the overall dialog quality (cf. Section 6.3.1) and the overall profile quality (cf. Section 6.3.2) separately, we also wanted find out whether there are correlations between dialog quality and profile quality. Such a correlation may be, for example, whether a dialog that was perceived very positively by the user also led to a high quality profile (i.e. a profile in which no or only few errors occurred).

For this purpose, we investigated – for each profile and corresponding dialog – correlations between the two **dialog criteria**

- total number of questions asked during the dialog,
- overall positive or negative “quality” of the user questionnaire (based on the judgment of the user about this dialog)

and the two **profile criteria**

- total rate of GermaNet errors that occurred for the profile,
- total rate of errors caused by other LINGUINI components for the profile.

The **total number of questions** varied for the dialogs due to the system’s functionality of ending the dialog dynamically according the contingent of fallback questions that has been used already. In average, 25 questions were asked per dialog.

The **overall positive or negative “quality” of the user questionnaire** was calculated as follows. Each answer possibility in the questionnaire was assigned a numeric value, which expresses whether the user’s judgment for this question was positive or negative (e.g. 4 for

“completely likes to use the system in the vehicle application scenario”, 3 for “very much likes ...”, and so on, until 0 for “not at all likes ... “). For each questionnaire we calculated the sum of these values, which indicates whether the questionnaire expresses a rather positive or rather negative overall opinion of the user with respect to LINGUINI.

These criteria were compared with the **total rate of GermaNet profile errors** and the **total rate of profile errors caused by other LINGUINI components**. The latter comprise all minor known errors, new errors/design problems, and also missing additional functionality.

Figure 6.16 shows the results of this comparison. The ID of each profile/questionnaire is indicated on the x-axis. The numbers indicated on the y-axis refer to different things for the individual curves: 1) the number of questions asked, 2) the number of the “subjective questionnaire quality”, and 3) the number of errors per profile.

The **topmost curve** in Figure 6.16 shows the “subjective quality of the questionnaire”, i.e. the overall positive or negative opinion of the user about LINGUINI. This opinion is better, if the value in the diagram is higher and worse if the value is lower. The **curve in the middle** indicates, how many questions have been asked in the dialog for a specific profile/user. The **bottom curve** indicates the LINGUINI component errors and the **dots at the bottom** indicate the number of GermaNet errors for this profile. For an interpretation of these results see the next section.

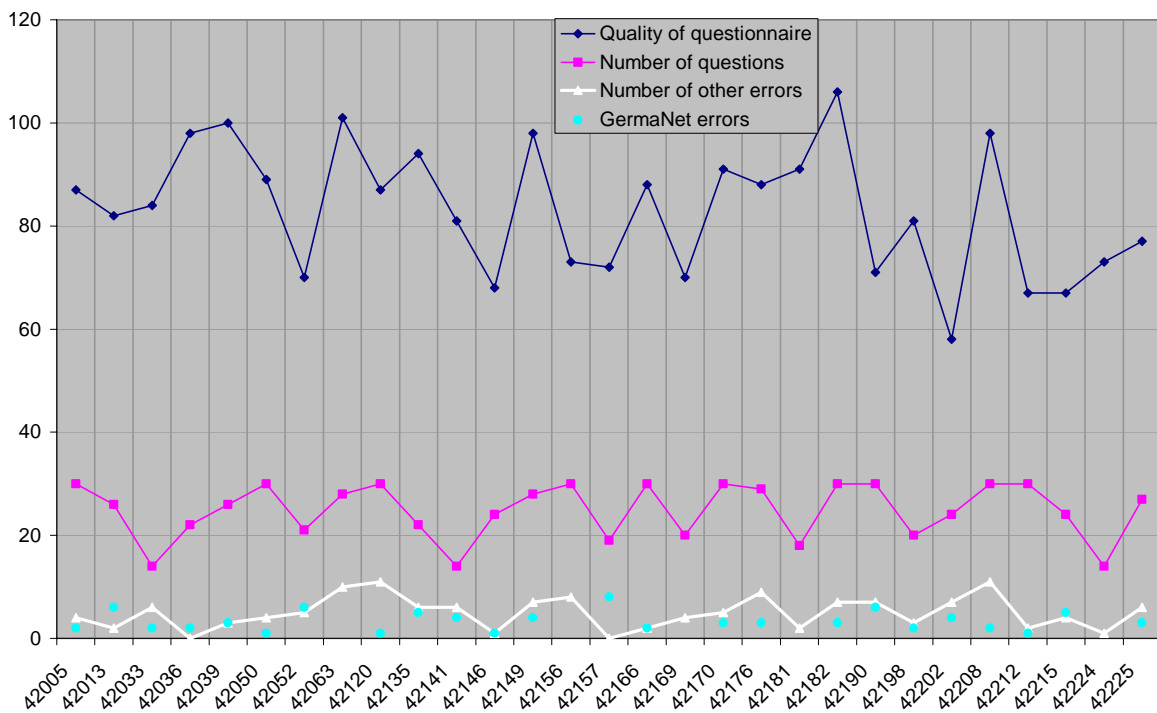


Figure 6.16: Correlations between user judgment, dialog length, and profile errors due to GermaNet and other system components

6.3.3 Summary of Results and Discussion

This section briefly summarizes, discusses, and interprets the most important results of the evaluations presented in Section 6.3.1 and 6.3.2.

Dialog User Study

With respect to the questions formulated in Section 6.3.1.5, the results of the user study can be interpreted as follows:

1.1 Emotions of the users: The feelings of the subjects were rather positive (fun, excitement) with respect to the interaction. Negative emotions (boredom or annoyance) occurred rarely, and the majority of users found it quite or completely interesting to talk with LINGUINI. The free text answers given by the users to the question as to what they particularly liked about LINGUINI (Q32) have shown that especially the dialog adaptability and capability to find related topics were major reasons for this positive judgment.

1.2 Adequacy of dialog length: About 50-60 % of the subjects perceived the dialog length as adequate for the task of generating personalized recommendations and sufficient in order to get to know their interests. About 40 % of the subjects found that a longer dialog would be better for this task. This may lead to the conclusion that users accept a lengthy dialog for this task and that even an increase in the length of dialog in the future would not necessarily be rejected by the users.

1.3 Application and usage situation: The results show that for most users the motivation to reveal profile information is higher if they can do something in parallel (Q26). The results for Q28 (can users imagine further everyday situations which would be suitable for conducting such a dialog) show that many users would prefer to do some manual task in parallel, bridge waiting periods, or conduct the dialog while they travel or pursue some activity outside their home. The results further show that the need for a profiling method with a speech interface that can also be used at the desktop computer at home is clearly given for a quarter of the subjects (Q29). For most users the motivation to conduct a profiling dialog with LINGUINI was higher on a longer highway trip than if they had to do it at home at their computers (Q27). This is substantiated by the result that three quarters of the users would like to use LINGUINI especially in the vehicle scenario for getting sightseeing and event recommendations (Q25). These results clearly argue for a speech dialog-based acquisition method as presented in this thesis, as opposed to other acquisition methods that require typed input and the undivided attention of the user. The profiling method presented in this thesis gives more freedom to the users because it enables them to provide profile information while doing something else, which is obviously very much appreciated by users.

2.1 Importance of adaptability: The study shows that the capability of the system to adapt to the user's interests is a very important and highly appreciated property of a profiling system, which about 90 % of the users would prefer over a standardized or non-adaptive system (Q11, Q13). This result substantiates our assumption that the development of adaptive profiling techniques is an important step towards increased user acceptance and cooperation and has in fact the potential to finally substitute standardized, non-adaptive profiling methods in many areas in the future.

2.2 Adaptation to interests mentioned by the user: The system was perceived as adaptive by most users, but the degree of adaptability and hence user satisfaction can still be increased in the future (Q1, Q2). From the answers given in Q7 (talk longer about individual interests?) and Q8 (asked enough detail questions?), we may conclude that the satisfaction of most users might probably be increased by allowing the system to talk longer about individual interests by asking more details about them. The results show that users, in general, would like to talk about specific interests longer and would prefer fewer topic changes. This can be achieved by changing the settings of the parameter which is responsible for the number of deepening questions asked about one interest in the DM (cf. Section 5.4.2.4).

2.3 The system's suggestion of related topics: The results show that the majority of users perceived the newly suggested topics as reasonable (Q6) and could understand why the sys-

tem suggested them as related to their interests (Q5). About half of the users found that the suggested topics were frequently or very frequently thematically related to their interests (Q4). This result shows that LINGUINI is already able to find related topics, but that this capability can be further increased in the future. This is supported by the lack of homogeneity of the answers given for Q2 and Q3, where we tried to elicit whether users found the suggested topics particularly interesting or inspiring. It also has to be taken into account, however, that problems perceived by the users during the dialog may influence their answers to rather generally formulated questions such as Q2 and Q3.

3.1 Errors: Although the system made mistakes occasionally (Q33), these were not perceived as very irritating (Q34). This means that with some minor improvements of the system, a sufficient level of reliability and proper functioning can already be reached in order to apply LINGUINI in practice.

3.2 Style of questions and answers: The formulation of the questions was perceived as quite good by most users (Q14, Q15), but further improvements could probably be achieved with the integration of a morphological question generation component (cf. Section 3.8.2). Since we use question frames with variable slots, a certain “repetition effect” cannot be completely avoided. For the answers, the results indicate that the mixture of free answers and answer choices has been perceived as already quite good and well-balanced in LINGUINI, but slightly more free answers would be desired by the users (Q20, Q21). This result also substantiates our hypothesis that users are happier with a profiling system which gives more freedom to them by allowing natural language answers as compared to a system which only offers pre-defined answers.

3.3 Improvements: From the free comments given by the users in Q35 (ideas about future improvements and desirable functionalities) we may infer what kinds of improvements would be mostly appreciated by the users. Taken together, these are: taking into account the individual semantics of an interest for selecting deepening questions more specifically, enabling users to select topics and influence the dialog flow, improving fallback questions, giving more feedback to the users and enabling them to see their profile, and a higher fault tolerance for the input. Some of these aspects have been considered in our concept already and have merely not as yet been implemented, such as the visualization of the created profile for the user. Others are interesting topics for future work (cf. Section 7.2). Overall, we may conclude that users desire a high degree of transparency and control. This substantiates our claim made in Chapter 2, that transparency is a key factor in user profiling.

Overall result of the user study: The user study has shown that most users enjoyed using LINGUINI and would like to use it again for getting personalized recommendations in a real application scenario. Users especially appreciate the fact that LINGUINI can find new topics related to their interests autonomously and consider adaptability of the dialog a very important feature. From the results, we can conclude that users actually prefer an adaptive profiling method to a standardized one. A speech dialog-based method as suggested in this thesis is particularly desirable for most users because they can pursue some other activity in parallel. With this study, we have shown that LINGUINI – already in this first prototype – has reached a high level of user acceptance and has demonstrated various adaptive capabilities. These adaptive capabilities, of course, still have potential for some future improvements. We have shown that it is possible to realize an adaptive dialog system that meets our requirements and that it would be very promising and desirable from the point of view of the users to further develop this approach for practical use.

Profile Assessment

The results for the criteria we investigated in the profile quality and coverage assessment (cf. Section 6.3.2.2) can be interpreted as follows:

Errors caused by external system components: The main problems caused by **GermaNet** were due to the fact that it does not yet cover all words of the German language and that it is not intended to contain proper names, except for some very frequently used ones (e.g. “Mittelmeer (Mediterranean)”). For LINGUINI, this had the consequence that 79 interests entered by users could not be found in GermaNet and could hence not be further asked about. This often leads to an earlier termination of the dialog, because it has used up all its fallback questions. The number of meaningful questions that can be asked during the dialog could be increased and the intended dialog length could be reached more often, if GermaNet’s coverage could be extended and if, in addition, a way could be found to semantically relate proper names to other words belonging to the German standard vocabulary. As opposed to this, the few misinterpretations caused by **TreeTagger** did not have a strong impact on the profile construction and management.

Errors caused by the other LINGUINI components: Most of the problems that were caused by other system components were related to missing, wrong, or overwritten attribute values and only few to missing or double profile nodes. As pointed out above, some of them were not addressed in the implementation in favor of more important functionality or problems that had a greater impact on the course of the dialog. However, most of them only affect minor changes that can be achieved with not too much effort (cf. Section 6.3.2.2).

Missing additional functionality with impact on profile quality and coverage: The three types of missing functionalities we found can be realized by integrating a disambiguation component, by manually cleaning up the Semantic Classes Lexicon, and by extending the Attribute Extraction Rules in the LA component to cover further linguistic phenomena. However, they do not frequently cause problems for the profile management (0.6 errors per profile in average). Hence, the implementation priority of the additional functionalities has been judged by us correctly in advance: they would be “nice to have”, but their absence in the prototype did not cause major problems for the profile management.

Correlations between positive/negative questionnaire quality, dialog length, and profile errors: First, the fact that 25 questions were asked per dialog is a rather good result, considering that, per dialog, the system could ask 30 questions at most and could terminate at any smaller number of questions in case its contingent of fallback questions was used up.

The most interesting result, however, is that Figure 6.16 in Section 6.3.2.2 shows that there is in fact a **correlation between questionnaire quality and dialog length**: all dialogs with a length of about 20 questions and less were judged as less positive by the users in the questionnaires. A dialog length that is shorter than 20 questions is mainly due to the fact that the system could not relate the user interests to interests in the TGMOD or did not find them in GermaNet. In these cases, the contingent of fallback questions is used up earlier and the dialog terminates earlier. This makes clear why these users felt less understood by the system and gave a rather middle-rate or negative overall judgment. Vice-versa, the questionnaires with a very positive overall judgment almost all refer to dialogs with a clearly longer duration. This can be interpreted such that the system understood the interests better and could ask more meaningful questions before having to access the contingent of fallback questions, which is reflected by the positive overall impression of the users. Taken together, this means that for adaptive dialogs the best interaction is not necessarily the shortest. Users are more willing to accept a longer dialog which is more adaptive than a shorter dialog that uses many general fallback questions.

Figure 6.16, moreover, shows that the course of the curve “number of other errors” (i.e. all errors except for GermaNet errors) is very similar to the course of the curve “number of questions” (i.e. the dialog length). This means that the **number of errors that occur in the profile increases with the dialog length**, which is what we would expect. These profile errors, which are not directly visible for the user, have to be distinguished from errors occurring during the dialog and which are directly noticeable by the user. This explains, why the **questionnaire quality does not correlate with the profile error rate** (produced both by GermaNet and other LINGUINI components) in general.

Overall result of the profile assessment: The profile assessment showed that in general the profiles have been created and updated to a satisfactory degree. LA and PM interacted correctly with GermaNet and TreeTagger in most cases and the profiles created during the user study capture most of the information entered by the users. The errors that occurred in the profile construction and management were of only a few different types and were mostly known errors. Most of them can be fixed with not too much effort, for example, by extending the linguistic coverage of the LA or by integrating the disambiguation component described in Section 3.6.3. Moreover, it would be desirable if GermaNet’s coverage could be extended and errors in the GermaNet XML files could be removed. It would also be helpful, if a method could be developed for accessing and relating semantic information about proper names. This is not an easy problem to solve, however. Finally, an interesting result is that longer dialogs were often judged more positively by users than shorter dialogs. Hence, brevity should not be the primary goal in adaptive profiling dialogs and our current upper limit of 30 questions might even be increased.

6.4 Summary

This chapter discussed the implementation and evaluation of the concept for our profiling approach, which we presented in the previous chapters. The design of the approach taken in LINGUINI has been based on an **empirical user study about preferred input formulations** presented in Section 6.1. The most important result of this study was that users prefer to enter short natural language interests of a rather simple syntactic structure and that the data collected in three different scenarios show a high degree of homogeneity in this respect. It was on this result that we especially based the design of our Language Analysis component (for which a shallow parsing technique is sufficient because of the simple input structure) and our Profile Model (which represents user interests as complex attribute-value pair structures).

Since we have implemented the main components and main system functionality exactly as presented in Chapters 4 and 5, we described only major implementation deviations from this concept and further implementation-specific aspects in Section 6.2. The aim of the **prototype implementation** was to enable test users to conduct a complete profiling dialog during a user study and to have a profile created and updated for each of them. Therefore, the prototype had to comprise the main functionalities of both the dialog management and the profile management part, i.e. we implemented the complete system (rather than merely demonstrating general feasibility by means of some partial implementations). We implemented the system in JAVA and XML such that it runs on a Web-server and can be accessed by test users via the Internet. Intensive component tests, which we conducted prior to the user study, were not only supposed to ensure the proper functioning of the components, but also helped us to find the most suitable parameter settings for the user study.

In the third part of this chapter, we described the **final evaluation** of the implemented prototype. It consisted of a user study about dialog adaptability, user acceptance, and the state of development of the prototype and a succeeding profile quality and coverage assessment. The main result of the **user study** was that users liked to talk with LINGUINI and had many ideas

for situations in which it could be used in practice. According to this study, the main advantages of LINGUINI seem to be its adaptability to individual users and the fact that it can be applied by means of a speech interface at different locations or in parallel to some other activity. The main result of the **profile assessment** was that no major errors occurred during the creation and update of profiles in the user study. With this assessment, we could also gain a clear idea about and helpful suggestions for possible future improvements of the profile management.

In Section 1.2, we mentioned a user study by [McNee et al. 2003] that has shown that a personalized profiling method in which users have the possibility to enter information more freely increases user loyalty and satisfaction and even leads to the acceptance of a longer profiling process. Our results clearly confirm this observation, especially that users would even accept a longer profiling process with LINGUINI. Overall, we can conclude that the user acceptance and willingness to reveal personal information increases with the capability of the system to adapt to users individually and its applicability in different (e.g. mobile) situations, in which users can pursue other activities in parallel.

7 Conclusions and Prospects

In this final chapter, we summarize the most important results and main contributions of this thesis. Furthermore, we discuss interesting areas for future research.

7.1 Summary of Contributions

The **starting point** of this thesis was the observation that existing explicit profiling methods suffer from severe drawbacks which limit their utilizability in practice, especially in mobile scenarios or in situations that require speech interaction. A main drawback of explicit profiling methods is that they typically cannot adapt to individual users or capture new, not predefined information a user wants to enter. These methods leave the users less freedom in describing themselves and influencing the type of information acquired about them. It is exactly this freedom however, which has been shown to be capable of increasing user satisfaction and loyalty (cf. e.g. [McNee et al. 2003]). Another main problem is that the most typical explicit profiling approach, online questionnaires the user has to fill in, is not suitable for many mobile situations, e.g. in-car use. Furthermore, existing approaches that acquire information about users and that allow for speech interaction are mainly bound to specific applications. This means, if a user profile is created at all – which is not the case for many of these approaches, because they provide the user with application-dependent information (e.g. recommendations) right away and discard what the user has entered after the session – then it cannot be used for other applications.

The **aim of this thesis** was therefore to develop an alternative explicit profiling method for the acquisition of personal interests of users with the following main characteristics:

- Besides being suitable for typed input, the method should be applicable in situations where only an interaction via speech is possible, such as in some mobile situations, for example. This argued for a profiling method based on a natural language dialog.
- The dialog should be able to adapt dynamically to the interests of an individual user. The method should also make it possible to acquire new information that has not been predefined in the profile and hence lead to the creation of truly personalized profiles.
- The profile should relate user interests semantically and provide a knowledge representation which can be interpreted by various applications. The profile size and structure should adapt automatically to the different contents stored for each user.

For this thesis, we developed a dialog management and a profile management approach which possess the above mentioned characteristics. In order to integrate the two parts, we realized the LINGUINI user profiling framework.

We started the **development of our solution** with an empirical user study that investigated how users prefer to formulate their interests in natural language. On the basis of the results of this study, our Profile Model and the Language Analysis component were designed. For representing semantic relations between interests in the user profile we use the semantic

graph structure of GermaNet. For analyzing the natural language user input, we developed a Language Analysis component which is supported by the NLP tool TreeTagger and its associated chunker. The Language Analysis integrates a set of Attribute Extraction Rules, which we have developed for creating semantically complex interest nodes, consisting of attribute-value pairs, from the user input. The profile construction and maintenance is accomplished by a Profile Manager component.

For the dialog management part, we employ a sociological Target Group Model which helps to select meaningful questions adaptively for each user, on the basis of the interests the user has mentioned. Our question selection approach takes into account various types of semantic relations between the interests in the user profile and the interests in the Target Group Model. We have developed a method for calculating the semantic similarity between user profile and target groups and between the individual interests contained in them. In particular, our dialog approach is capable of suggesting new topics that are related to topics mentioned by the user and is configurable for the needs of different applications. We extended the original sociological Target Group Model by means of further interests taken from the user study conducted at the beginning of the project. This has the effect that, in general, user profiles now show a higher similarity to one or several target groups and hence better questions can be generated.

The approach has been complemented with a **comprehensive implementation** of the main system components and functionality. We evaluated this prototype implementation by means of an extensive dialog user study and profile quality assessment. The results of this evaluation show that – even with this first implementation – LINGUINI has already reached a high level of user acceptance. It has also proven its adaptive capabilities by suggesting related topics that were much appreciated by the users. The results clearly indicate that users would very much like to use our profiling system in different (e.g. mobile) application scenarios.

The **main contribution of this thesis** is a profound concept for an explicit profiling approach that comprises all of the above mentioned characteristics as well as a comprehensive prototype implementation, which is platform independent, constitutes a solid basis for a future software system that can be practically applied, and has been evaluated by means of a final extensive user study.

The most important innovative properties of our solution are:

- The comprehensiveness of our approach to cover not only some but all of the above mentioned characteristics (other existing approaches cover only parts of them)
- The ability to acquire new information that has not been predefined in a user model before and to store this information in a user-individual, semantically structured graph
- The ability of the Dialog Manager to find and suggest, for each user individually, new topics that are semantically related to interests of this particular user

A sample application scenario, in which the driver of a car can talk about her personal interests in order to get sightseeing and event recommendations, has been sketched in order to illustrate the applicability of our approach in situations that require a spoken language interaction. However, a key feature of our approach is that our profiling system can be used in various application scenarios – for speech and typed input/output – and that the created user profiles can be applied for different types of adaptive systems, e.g. systems recommending completely different types of products or information in a personalized manner.

Another contribution of this work is that it has tested the practical suitability of the present state of the linguistic resource GermaNet – which is still under development – in a real application (i.e. a profiling system) and has contributed to eliciting advantages and shortcomings of this resource that might be improved in the future.

7.2 Future Work

In this thesis, we have presented an approach to explicit user profiling, which has focused on the management of a user-adaptive natural language profiling dialog and on the management of semantically structured user profiles. Our main contribution was to realize this approach with its specific properties comprehensively and from scratch as described in Section 7.1, which has not been done in this way before by any other explicit profiling approach. For this reason, we could not investigate some further interesting aspects that were out of the scope of this aim. In this section, we describe some of these research topics related to our work that would be worth investigating further in the future:

- User interfaces
- The visualization of user profiles
- Extension of the core functionality of the LINGUINI prototype
- Exchange of user information and privacy
- Applications and further evaluations

7.2.1 User Interfaces

As mentioned above, our focus was on the underlying main functionality of the profiling system. A topic which could therefore not be addressed in depth in this thesis is that of possible user interfaces (UIs). A particular asset of LINGUINI is that it can be employed both with a speech interface in mobile scenarios and with UIs that permit typed input and graphical visualization for the use on Web platforms, for example. Typically, UIs depend to a great extent on the type of application and usage situation. We have therefore kept the interface between Dialog Manager and UI as simple and flexible as possible. Since our test UI for typed input (cf. Sections 5.4.1.2 and 6.2.3.6) contains no “knowledge” about internal system processes, it can be easily exchanged by any UI that is suitable for a specific application.

A state-of-the-art approach for a typed input/output UI of an Internet platform, for example, would be to employ an (animated) “virtual character” who conducts the dialog with the user. [Pandzic 2001] discusses different technical possibilities to realize such virtual characters graphically and presents various approaches. Figure 7.1 shows an example of a virtual character that is given a specific identity by using an image of a person:



Figure 7.1: An example of a virtual character (source: [Pandzic 2001])

Possibilities for speech UIs have been discussed in Section 3.2. It would be interesting to investigate – for both typed and spoken input/output – what kinds of UIs are particularly suitable for a profiling dialog (as compared to other dialog types) and what requirements these

UIs have to fulfill for this kind of application. Since our profiling dialog adapts to the individual user, it would also be desirable to use it in connection with UIs that can be personalized to a certain degree according to the wishes of the individual user (e.g. with respect to the appearance, personality, or wording of the virtual character). Moreover, virtual characters may be employed that are able to adapt their gestures and facial expressions as well as their actions to the course of the dialog. For further information on animated, life-like virtual characters e.g. cf. a survey given in [Rist et al. 2003].

Another kind of UI that has not yet been integrated into LINGUINI is the UI to the Target Group Design Tool (cf. Section 5.3.1.1). The typical user of the Target Group Design Tool is an application designer who wants to create a Target Group Model for LINGUINI that represents typical user groups of her specific application. Our prototype implementation already provides part of the functionality that would be needed for such a tool but to realize the tool as such was not in the focus of our work. During our work, we formulated some requirements for such a UI, however, which we have presented in Section 5.3.1.1 (e.g. a simplified and customized visualization of the GermaNet graph structure and support for the application designer in selecting paths representing specific meanings of interests). Further requirements would have to be investigated in case a Target Group Design Tool is to be realized for LINGUINI in the future.

7.2.2 The Visualization of User Profiles

In Section 2.1.1.3, we mentioned that transparency is a key factor for profiling systems. This view is substantiated by our final system evaluation (cf. Section 6.3), in which users expressed the desire for more transparency with respect to the data stored about them and for feedback as to whether the system understood them correctly. Visualizing the user profile is essential for preserving and increasing the user's trust and cooperation. In our concept, we have already considered this aspect by suggesting that the user profile should be visualized for the users and should be accessible to them at anytime in order to make changes. In the prototype implementation, we have not yet integrated a sophisticated way of visualizing user profiles, however, because this was not a part of the main profile management functionality, but is rather related to the topic of UIs (cf. Section 7.2.1) and should also involve an investigation of usability aspects. At the moment, LINGUINI only offers the standard database view on the underlying XML representation of the user profile.

Since our user profiles are based on the complex structure of GermaNet, it is important to visualize the contents of a profile in a clearly laid out manner that does not overstrain the user. A possibility would be to represent the XML tree structure of the profile similar to the hierarchical structure of a file management system (cf. Figure 7.2 below).

In this hierarchical structure, the user can open or close profile nodes in order to see or hide subpaths of the tree. The user should also be able to add or remove information from interest nodes (cf. option "edit node" in the box in Figure 7.2). This option could also be used to transform any node in a path into an interest node by specifying attribute values for it. With the option "remove all interest information from this node" the user can transform an interest node back into a normal intermediate node (cf. Section 4.4.2), which no longer represents a user interest. Furthermore, an option "create new interest node under this node" could enable the user to insert completely new interests into the profile. However, this option would have to be supported by automatically created suggestions for legal paths for a new interest (taken from GermaNet) in order to keep the entire profile structure consistent.

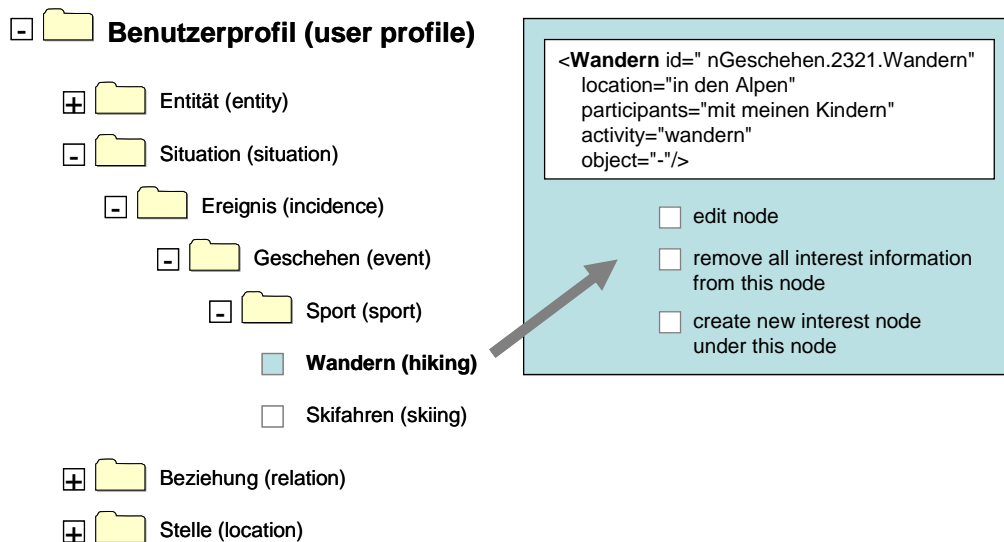


Figure 7.2: Visualizing the complex profile representation for the user

7.2.3 Extension of the Core Functionality of the LINGUINI Prototype

In the course of this thesis, we mentioned some aspects that were not in the focus of our work but for which our approach could be extended in the future. In the following, we summarize the most interesting possible extensions of the core system functionality from which the prototype could profit most.

1. Not yet implemented functionality: Some aspects have already been considered by us in the concept, but could not yet be integrated into the prototype implementation, because they exceeded the main functionality intended in LINGUINI. However, the system could be further improved by integrating these aspects.

First, this is the **word sense disambiguation** of user input with several possible meanings. Here, we would recommend starting off with the integration of the disambiguation component implemented by [Eiseler 2005] into the LINGUINI prototype and further improving this approach (cf. Section 3.6.3). For this purpose, the currently used GermaNet-API has to be extended as described in Section 6.2.3.2. A comprehensive ambiguity handling is a very interesting and non-trivial linguistic research topic, however, and would probably even exceed the disambiguation approach we have sketched in Section 3.6.3.

Secondly, an external **morphological tool** could be integrated into the Question Generation component. This would enable the generation of grammatically correct natural language output, in which, for example, word endings are inflected correctly. For demonstrating that questions with reasonable and correct contents can be generated, the simple question generation approach we have implemented was sufficient. In this approach, words that serve as variable values are inserted into the question frames without generating the correct word endings for these words. Sometimes this leads to slightly ungrammatical system output, but this output can be understood correctly nevertheless. Our approach allows, however, for the easy addition of a morphological tool, as described in Section 3.8.2.

A third possibility for extension is less interesting from a research point of view, but would further improve the usability of the system: our implementation does not yet integrate a **spell checking functionality**. Therefore, the user input has to be free of typing errors in order to be processed correctly by the system. While typing errors and the capitalization of words are not a problem for a spoken dialog interaction, they have to be handled for UIs with typed

input. For this purpose, an off-the-shelf spell checker tool, which removes typing errors from the user input automatically, can be employed.

2. Further ideas derived from the system evaluation: The user study with which we evaluated the prototype brought up some interesting points that would be worth investigating in the future. We will only discuss two particularly interesting ideas here and for further possible extensions refer to Section 6.3.1.5 (“3.3 Improvements”) and Section 6.3.2.2, in which we have described them in detail. In Section 6.3.2.2, we have, in particular, also discussed possible extensions of the Language Analysis and the Semantic Classes Lexicon.

The first aspect we would like to mention here is **initiative** during the dialog. In our approach, only the system may take the initiative by asking questions. The users do not have the possibility of asking questions themselves or taking other direct influence on the course of the dialog. In Section 4.2.1.2, however, we have explained that users may *indirectly* influence the dialog by mentioning new topics (i.e. interests) to which the dialog tries to adapt in turn. This strategy has proven sufficient for acquiring user interests in our approach and our study has shown that the degree of user acceptance for such a dialog is in fact satisfactory. We also wanted to realize a robust dialog approach in this first prototype, which is technically easier for a single-initiative approach in which less possible dialog situations have to be handled. For adaptive profiling dialogs, it would also be interesting, however, to investigate further possible dialog strategies. Another strategy might, for instance, allow both system and user to take the initiative and to use speech acts (e.g. question, confirmation, rejection, etc.) more flexibly. Since some users have uttered the desire to take more influence on dialog control, it would be interesting to see whether a mixed-initiative approach would in fact increase user satisfaction in a noticeable way.

A second aspect is the extension of our question selection approach for **pragmatical information** as mentioned in Section 5.4.2.4. Users noticed that LINGUINI sometimes asks deepening questions about specific attributes (e.g. participants) in connection with an interest for which this does not make much sense (e.g. “reading”). It would be desirable to find a way of determining in which case a specific attribute question really makes sense and in which case it should rather be avoided. From a linguistic point of view, this is a difficult problem which would require an in depth investigation and treatment that would constitute a larger research work of its own. The semantics of the verbs in the user input and of the other words that occur in their sentence environment would have to be taken into account. In particular, the restrictions verbs impose on their environment would have to be considered. But this would not be sufficient. A solution should also take into account “pragmatical” or “world knowledge” about user interests that exceeds the mere property of a verb to predetermine the words and phrases that may structurally occur in its environment.

7.2.4 Exchange of User Information and Privacy

In Section 1.3, we formulated the requirement that our profiling approach should lead to user profiles that are **exchangeable** and can be **reused by various applications**. With our approach we have provided the basis for this. In Section 2.1.2.4, several application types were described which can make use of the interest information acquired by LINGUINI. As opposed to many existing approaches (e.g. for acquiring preferences for eating out on a specific day in a specific area), the interests acquired with LINGUINI are not bound to a particular application. The XML-based representation of interest profiles further supports their exchangeability. What we could not address in the scope of this thesis, however, is the question of how the exchange should best be accomplished technically and which infrastructure is necessary for it. We assume that the user profiles are located on a central server which can be accessed by different applications and by the user for adding and modifying profile information (cf. Section

2.1.2.3). The realization of a technical framework for the exchange of profile information, however, has to be investigated in more depth in some future work. For this purpose, we also refer to existing work in this area, for instance [Fink and Kobsa 2000, Kobsa 2001, Koch 2002, Koch and Möslein 2003].

A topic related to the exchangeability of profile information is the **combination of different profiling methods**. It would be interesting to investigate how LINGUINI could be combined with other profiling methods and profile information acquired by other applications. For example, both implicit and explicit user feedback (cf. [Kuflik et al. 2005]) about the adequateness of recommendations generated by an application on the basis of the LINGUINI profile information could be taken into account. The LINGUINI dialog method could moreover be combined with an explicit questionnaire for the acquisition of basic personal information of simple structure (e.g. name, age, gender, etc.), which is often needed for specific applications. It could also be investigated whether the interest information acquired by LINGUINI could be complemented by implicitly collected information about topics users find interesting. These topics could be derived, for example, from the user's browsing behavior when visiting Websites of interest.

Furthermore, when making profile information accessible for multiple applications and perhaps also for other users, the issue of **privacy** has to be considered. In Sections 2.1.1.3 and 2.2.2.2, we pointed out that privacy is an important topic for all user profiling approaches. We have outlined that, in our opinion, user modeling must direct its view towards profiles that are owned and managed by the user rather than by the application. It would be interesting to investigate whether LINGUINI could be integrated with such an approach. In explicit profiling approaches users are directly asked to enter personal information. The awareness of users about their personal data being collected and stored is consequently higher for explicit approaches than for implicit ones. Therefore, explicit approaches have to face privacy concerns of the users more urgently in order not to lose their trust, especially if the information acquired is very personal. Giving users the chance to define access rights for specific parts of their profiles, which may only be accessed by specific applications or specific other users, may be one solution towards this direction (cf. [Wörndl 2003]).

7.2.5 Applications and Further Evaluations

As mentioned earlier, we evaluated the main functionality we have realized in our prototype by means of a comprehensive user study, with which we could show that we have in fact reached our defined goals. A major outcome of this evaluation was that users had many ideas for possible application situations for LINGUINI and that they would in fact like to apply it for profiling in order to profit from adaptive services in turn, especially in the car scenario.

In our user study, we could only evaluate our approach by means of typed input/output in a situation in which users access the profiling system from their desktop computers. It would also be interesting to evaluate LINGUINI in a mobile scenario with an integrated speech interface. It would also be possible to evaluate it in a **set of different scenarios** in order to find out for which applications it is particularly suitable. For finding further application scenarios, the suggestions provided by our subjects (cf. Section 6.3.1.5 "1.3 Application") can also be taken into account.

Furthermore, it would be interesting to evaluate LINGUINI with **other Target Group Models**. We have already tested LINGUINI with two models: the original Target Group Model we have developed according to the sociological model of [Schulze 1992] and an extended version of this model, to which we have added further user interests derived from the user data collected in our first study (cf. Section 6.2.3.3). With the extended model, more meaningful questions could be asked during the dialog, which shows that the kind of Target Group Model, in fact, influences the course of the dialog. Therefore, it would be interesting to

investigate this influence in more detail in the future. Different Target Group Models could be created, for instance, that represent typical user groups of specific applications. The dialogs the system conducts on their basis could be compared and analyzed. On the basis of this comparison inferences could be made as to what aspects should be considered when designing a Target Group Model and what exact impact its contents and size have on the course of the dialog and the selection of questions. The results could be summarized as guidelines for the application designers who want to create their own Target Group Models.

Appendix A:

A Sample Target Group: The Extended Niveaumilieu

This appendix contains the complete XML representation of the target group “Niveaumilieu” as it is currently used in the LINGUINI prototype. This is one of the four target groups we use in our Target Group Model (cf. Section 5.3.1.2). The target group has been extended to include additional interests taken from our empirical user study conducted at the beginning of the project (cf. Section 6.1).

```
<?xml version="1.0" encoding="UTF-8"?>
<Profilinstanz id="30001">
  <NOT_FOUND_IN_GN_NOUNS/>
  <Situation id="x.0.Situation">
    <Ereignis id="x.0.Ereignis">
      <Geschehen id="x.0.Geschehen">
        <Vorfall id="x.0.Vorfall">
          <Erlebnis id="x.0.Erlebnis">
            <Reise id="x.0.Reise">
              <Seereise id="x.0.Seereise">
                <Kreuzfahrt id="x.0.Kreuzfahrt" object="Kreuzfahrten" weight="35"/>
              </Seereise>
            </Reise>
            <Ausflug id="x.0.Ausflug">
              <Spaziergang id="x.0.Spaziergang" object="Spaziergänge" weight="57"/>
            </Ausflug>
          </Erlebnis>
        </Vorfall>
      </Geschehen>
      <Veranstaltung id="x.0.Veranstaltung">
        <kulturelle_Veranstaltung id="x.0.kulturelle_Veranstaltung">
          <Vorführung id="x.0.Vorführung">
            <Musikveranstaltung id="x.0.Musikveranstaltung">
              <Konzert id="x.0.Konzert" object="Konzerte" property="klassische"
                weight="41"/>
              <Oper id="x.0.Oper" object="Oper" weight="30"/>
            </Musikveranstaltung>
          </Vorführung>
        </kulturelle_Veranstaltung>
        <Fest id="x.0.Fest">
          <Volksfest id="x.0.Volksfest" activity="besuchen" dislike="kein"
            object="Volksfest" weight="21"/>
        </Fest>
      </Veranstaltung>
    </Ereignis>
    <Handlung id="x.0.Handlung">
      <Freizeitaktivität id="x.0.Freizeitaktivität">
        <Zerstreuung id="x.0.Zerstreuung">
          <kulturelle_Veranstaltung id="x.0.kulturelle_Veranstaltung">
            <Vorführung id="x.0.Vorführung">
              <Filmvorführung id="x.0.Filmvorführung">
                <Film id="x.0.Film">
                  <Themenfilm id="x.0.Themenfilm">
                    <Actionfilm id="x.0.Actionfilm" activity="ansehen" dislike="keinen"
                      object="Actionfilm" weight="35"/>
                  </Themenfilm>
                </Film>
              </Filmvorführung>
            </Vorführung>
          </kulturelle_Veranstaltung>
          <Fernsehveranstaltung id="x.0.Fernsehveranstaltung">
            <Quizsendung id="x.0.Quizsendung" activity="ansehen" dislike="keine"
              object="Quizsendung" weight="46"/>
          </Fernsehveranstaltung>
        </Zerstreuung>
      </Freizeitaktivität>
    </Handlung>
  </Situation>
</Profilinstanz>
```

```

        </Fernsehveranstaltung>
    </kulturelle_Veranstaltung>
</Zerstreuung>
</Freizeitaktivität>
<Handel id="x.0.Handel">
    <Handwerk id="x.0.Handwerk">
        <Handarbeit id="x.0.Handarbeit" activity="machen" dislike="keine"
            object="Handarbeit" weight="15"/>
    </Handwerk>
</Handel>
</Handlung>
</Geschehen>
<Prozess id="x.0.Prozess">
    <kognitiver_Prozess id="x.0.kognitiver_Prozess">
        <Wahrnehmung id="x.0.Wahrnehmung">
            <geistige_Wahrnehmung id="x.0.geistige_Wahrnehmung">
                <Kognition id="x.0.Kognition">
                    <Lernen id="x.0.Lernen">
                        <Ausbildung id="x.0.Ausbildung">
                            <Weiterbildung id="x.0.Weiterbildung" object="Fortbildung" weight="45"/>
                        </Ausbildung>
                    </Lernen>
                </Kognition>
            </geistige_Wahrnehmung>
        </Wahrnehmung>
    </kognitiver_Prozess>
</Prozess>
</Ereignis>
</Situation>
<Stelle id="x.0.Stelle">
    <Aufenthaltort id="x.0.Aufenthaltort">
        <Theater id="x.0.Theater" object="Theater" weight="30"/>
    </Aufenthaltort>
</Stelle>
<Entität id="x.0.Entität">
    <Objekt id="x.0.Objekt">
        <natürliches_Objekt id="x.0.natürliches_Objekt">
            <Lebewesen id="x.0.Lebewesen">
                <natürliches_Lebewesen id="x.0.natürliches_Lebewesen">
                    <höheres_Lebewesen id="x.0.höheres_Lebewesen">
                        <Tier id="x.0.Tier">
                            <Gebetier id="x.0.Gebetier">
                                <Chordatier id="x.0.Chordatier">
                                    <Wirbeltier id="x.0.Wirbeltier">
                                        <Säugetier id="x.0.Säugetier">
                                            <höherer_Säuger id="x.0.höherer_Säuger">
                                                <Huftier id="x.0.Huftier">
                                                    <Unpaarhufer id="x.0.Unpaarhufer">
                                                        <Pferd id="x.0.Pferd" object="Pferde" weight="29"/>
                                                    </Unpaarhufer>
                                                </Huftier>
                                            </höherer_Säuger>
                                        </Säugetier>
                                    </Wirbeltier>
                                </Chordatier>
                            </Gebetier>
                        </Tier>
                    </höheres_Lebewesen>
                </natürliches_Lebewesen>
            </Lebewesen>
        </natürliches_Objekt>
    <Nahrung>
        <flüssiges_Nahrungsmittel id="x.0.flüssiges_Nahrungsmittel">
            <Getränk id="x.0.Getränk">
                <alkoholisches_Getränk id="x.0.alkoholisches_Getränk">
                    <Wein id="x.0.Wein" object="Wein" property="guter" weight="30"/>
                </alkoholisches_Getränk>
            </Getränk>
        </flüssiges_Nahrungsmittel>
    </Nahrung>
<Ding id="x.0.Ding">
    <Artefakt id="x.0.Artefakt">
        <Produkt id="x.0.Produkt">
            <Gerät id="x.0.Gerät">
                <akustisches_Gerät id="x.0.akustisches_Gerät">
                    <Musikinstrument id="x.0.Musikinstrument" object="Musikinstrumente"
                        weight="35">
                        <Tasteninstrument id="x.0.Tasteninstrument">

```

```

        <Klavier id="x.0.Klavier" object="Klavier" weight="35"/>
    </Tasteninstrument>
    <Saiteninstrument id="x.0.Saiteninstrument">
        <Streichinstrument id="x.0.Streichinstrument">
            <Geige id="x.0.Geige" object="Geige" weight="35"/>
        </Streichinstrument>
    </Saiteninstrument>
</Musikinstrument>
</akustisches_Gerät>
</Gerät>
</Produkt>
<Bauwerk id="x.0.Bauwerk">
    <Gebäude id="x.0.Gebäude">
        <Haus id="x.0.Haus">
            <Wohnhaus id="x.0.Wohnhaus">
                <Ferienhaus id="x.0.Ferienhaus" object="Ferienhäuser" weight="35"/>
            </Wohnhaus>
        </Haus>
        <Lokal id="x.0.Lokal">
            <Speisegaststätte id="x.0.Speisegaststätte">
                <Restaurant id="x.0.Restaurant" property="gehobene" object="Restaurants"
                    weight="55"/>
            </Speisegaststätte>
        </Lokal>
        <Ausstellungsort id="x.0.Ausstellungsort">
            <Museum id="x.0.Museum" object="Museum" weight="27"/>
        </Ausstellungsort>
    </Gebäude>
</Bauwerk>
<Kunstwerk id="x.0.Kunstwerk">
    <Musikstück id="x.0.Musikstück">
        <Lied id="x.0.Lied">
            <Schlager id="x.0.Schlager" property="deutscher" activity="hören"
                dislike="keinen" object="Schlager" weight="47"/>
        </Lied>
    </Musikstück>
</Kunstwerk>
<Transportmittel id="x.0.Transportmittel">
    <Fahrzeug id="x.0.Fahrzeug">
        <Landfahrzeug id="x.0.Landfahrzeug">
            <Radfahrzeug id="x.0.Radfahrzeug">
                <Automobil id="x.0.Automobil" activity="pflegen" dislike="nicht"
                    object="Auto" weight="18"/>
            </Radfahrzeug>
        </Landfahrzeug>
    </Fahrzeug>
</Transportmittel>
</Artefakt>
<Teil id="x.0.Teil">
    <Teilmenge id="x.0.Teilmenge">
        <Gruppe id="x.0.Gruppe">
            <dingliche_Ansammlung id="x.0.dingliche_Ansammlung">
                <Sammlung id="x.0.Sammlung" object="Sammlung" favorite_items="Münzen"
                    weight="15"/>
            </dingliche_Ansammlung>
        </Gruppe>
    </Teilmenge>
</Teil>
</Ding>
<Hilfsmittel id="x.0.Hilfsmittel">
    <Medium id="x.0.Medium">
        <Musik id="x.0.Musik" property="klassische" activity="hören" object="Musik"
            weight="41"/>
        <Druckwerk id="x.0.Druckwerk">
            <Zeitung id="x.0.Zeitung" activity="lesen" object="Zeitung" weight="26"/>
            <Buch id="x.0.Buch" activity="lesen" object="Buch" weight="32"/>
            <Schriftstück id="x.0.Schriftstück">
                <Manuskript id="x.0.Manuskript">
                    <Tagebuch id="x.0.Tagebuch" activity="schreiben" object="Tagebuch" weight="20"/>
                </Manuskript>
            </Schriftstück>
            <Zeitungsteil id="x.0.Zeitungsteil">
                <Kulturbeilage id="x.0.Kulturbeilage" activity="lesen" object="Feuilleton"
                    weight="20"/>
            </Zeitungsteil>
        </Druckwerk>
    </Massenmedium id="x.0.Massenmedium">
        <Fernsehen id="x.0.Fernsehen" activity="ansehen" dislike="kein" object="Fernsehen"

```

```

        weight="25"/>
    </Massenmedium>
    <Sprache id="x.0.Sprache" activity="lernen" object="Sprache" weight="33"/>
</Medium>
</Hilfsmittel>
</Objekt>
<kognitives_Objekt id="x.0.kognitives_Objekt">
    <Kategorie id="x.0.Kategorie">
        <Art id="x.0.Art">
            <Stil id="x.0.Stil">
                <Kunststil id="x.0.Kunststil">
                    <Musikstil id="x.0.Musikstil">
                        <Jazz id="x.0.Jazz" activity="hören" object="Jazz" weight="34"/>
                        <Unterhaltungsmusik id="x.0.Unterhaltungsmusik">
                            <Pop id="x.0.Pop" activity="hören" dislike="keine" object="Popmusik"
                                weight="0"/>
                        </Unterhaltungsmusik>
                    </Musikstil>
                </Kunststil>
            </Stil>
        </Art>
    </Kategorie>
</kognitives_Objekt>
</Entität>
<Beziehung id="x.0.Beziehung">
    <Kommunikation id="x.0.Kommunikation">
        <Gespräch id="x.0.Gespräch" object="Gespräche" property="gute" weight="19">
            <Diskussion id="x.0.Diskussion" property="politische" object="Diskussion"
                weight="19"/>
        </Gespräch>
        <Äußerung id="x.0.Äußerung">
            <Vorschlag id="x.0.Vorschlag">
                <Empfehlung id="x.0.Empfehlung">
                    <Reklame id="x.0.Reklame" dislike="keine" object="Werbung" weight="27"/>
                </Empfehlung>
            </Vorschlag>
        </Äußerung>
    </Kommunikation>
</Beziehung>
</Profilinstanz>

```

Appendix B:

The Final System Evaluation

B.1 The Questionnaire

This appendix contains the complete questionnaire we asked the users to fill in after they had conducted a dialog with LINGUINI in our final user study (cf. Section 6.3). The questionnaire is described in Section 6.3.1.2.

Fragebogen zur Benutzerstudie “Interessenserhebung durch Sprachdialog”

Nun möchten wir Sie bitten, Ihren Dialog mit dem System anhand einiger Fragen zu beurteilen.

Kommentare: Falls Ihnen zu einer Frage mehr einfällt, können Sie gerne jederzeit einen zusätzlichen Kommentar einfach unter die Frage schreiben.

Erklärung zum Datenschutz: Alle von Ihnen gemachten Angaben werden anonym ausgewertet, nicht an dritte Personen weitergegeben und nicht für kommerzielle Zwecke verwendet.

WICHTIG: Tragen Sie hier bitte als erstes die **Profil-ID** ein (die Nummer, die Sie sich notiert hatten):

Warum ist die Profil-ID für uns wichtig: Ihr Dialog mit dem System wird in einem Profil anonym gespeichert. Jedem Profil wird vom System automatisch eine Nummer zugewiesen, die Profil-ID. Damit wir später auswerten können, ob das System alle Informationen korrekt gespeichert hat, benötigen wir die jeweilige Profil-ID.

1 Anpassungsfähigkeit des Systems

1. Wie stark ist der Dialog auf die von Ihnen eingegebenen Interessen eingegangen?
 außerordentlich ziemlich mittelmäßig kaum gar nicht
2. Waren neue Themen, die das System vorgeschlagen hat, für Sie im Allgemeinen interessant?
 außerordentlich ziemlich mittelmäßig kaum gar nicht
3. Wie anregend fanden Sie im allgemeinen neue Themen, die das System vorgeschlagen hat?
 außerordentlich ziemlich mittelmäßig kaum gar nicht

4. Waren neue Themen, die das System vorgeschlagen hat, thematisch verwandt oder ähnlich zu Ihren eigenen?

sehr häufig häufig gelegentlich selten sehr selten

5. Manchmal bringt das System ein von Ihnen eingegebenes Interesse mit einem neuen Thema in Verbindung (z.B. „Da Sie X mögen, mögen Sie auch Y?“). Fanden Sie, dass zwischen Ihrem Interesse und dem neuen Thema im Allgemeinen ein nachvollziehbarer Zusammenhang bestand?

sehr häufig häufig gelegentlich selten sehr selten

6. Fanden Sie den vom System hergestellten Zusammenhang im allgemeinen sinnvoll?

sehr häufig häufig gelegentlich selten sehr selten

7. Hätten Sie sich gerne über einzelne Interessen länger unterhalten (auch wenn dadurch weniger Themen insgesamt besprochen werden)?

ja nein

8. Hatten Sie den Eindruck, dass generell zu viele oder zu wenige Detail-Fragen zu einzelnen Interessen gestellt wurden?

zu viele genau richtig zu wenige

9. Sollte das System mehr oder weniger weitere/neue Themen vorschlagen und Sie dazu befragen?

mehr genau richtig weniger

10. Hatten Sie das Gefühl, der Dialog wechselt zu häufig oder zu selten das Thema?

zu häufig genau richtig zu selten

11. Wie wichtig finden Sie es, dass sich das System in so einem Gespräch an Ihre persönlichen Interessen anpasst?

außerordentlich ziemlich mittelmäßig kaum gar nicht

12. Wie sehr sind Sie mit dem Maß, mit dem das System auf Sie persönlich eingeht, zufrieden?

außerordentlich ziemlich mittelmäßig kaum gar nicht

13. Würden Sie lieber einen Dialog führen, der nicht auf Sie persönlich eingeht, sondern nur Standardfragen in fester Reihenfolge stellt, die für jeden Benutzer gleich sind?

ja, Begründung:

nein

2 Systemvorgaben und Fragestil

Zu den Fragen

14. Zum jetzigen Zeitpunkt fehlt im System noch eine Komponente, die die richtigen Wortendungen erzeugt. Deshalb klingen die Fragen oft nicht ganz grammatikalisch. Fanden Sie es – davon abgesehen – im allgemeinen ansprechend, wie die Fragen formuliert sind (z.B. war die Länge der Frage angenehm, wirkte der Fragestil natürlich/eingängig, etc.)?

außerordentlich ziemlich mittelmäßig kaum gar nicht

15. Haben Sie die Fragen, die das System gestellt hat, inhaltlich verstanden?

sehr häufig häufig gelegentlich selten sehr selten

16. Der Dialog nennt hin und wieder konkrete Interessen und fragt, ob diese auf Sie zutreffen. Haben Sie sich durch die Tatsache, dass das System selbst auch ein paar konkrete Vorschläge macht, entlastet gefühlt?

außerordentlich ziemlich mittelmäßig kaum gar nicht

17. Hätten Sie sich eher noch mehr oder weniger solche konkreten Vorschläge vom System gewünscht?

mehr genau richtig weniger

Zu den Antworten

Das System lässt einerseits freie Antworten zu (wo der Benutzer in seinen eigenen Worten antworten kann). Andererseits gibt es Antwortmöglichkeiten (ja/nein, wichtig/weniger wichtig, etc.) vor, aus denen der Benutzer auswählen kann.

18. Fanden Sie es schwierig, Ihre Interessen als freie Antworten auszudrücken?

außerordentlich ziemlich mittelmäßig kaum gar nicht

19. Fanden Sie es schwierig, Ihre Interessen durch die konkreten Antwortmöglichkeiten auszudrücken?

außerordentlich ziemlich mittelmäßig kaum gar nicht

20. Sollte das System lieber mehr oder weniger freie Antworten zulassen?

mehr genau richtig weniger

21. Sollte das System lieber mehr oder weniger konkrete Antwortmöglichkeiten vorgeben?

mehr genau richtig weniger

3 Dauer des Dialogs

Das System hat die Aufgabe, sich ein möglichst genaues Bild von Ihren Interessen zu machen, damit es Ihnen in Zukunft möglichst gute Empfehlungen (z.B. für interessante Veranstaltungen oder Freizeitangebote) geben kann.

22. Finden Sie, dass die Dialogdauer für diese Aufgabe eher zu kurz oder zu lang war?

eher zu kurz genau richtig eher zu lang

23. Denken Sie, der Dialog hat lange genug gedauert, um ein ausreichend gutes Bild von Ihren Interessen zu bekommen?

ja nein

24. Falls Sie den Dialog vorzeitig beendet haben, was war der Grund dafür:

4 Anwendung

Stellen Sie sich bitte nochmals die folgende Situation vor:

Sie fahren im Auto (als Fahrer/in oder Beifahrer/in). Sie beschließen spontan einen Abstecher in eine nahe gelegene größere Stadt zu machen. Leider hatten Sie vor der Reise keine Zeit, sich über Freizeitmöglichkeiten, Veranstaltungen und Attraktionen in dieser Stadt zu informieren. Um die Stadt zu erreichen fahren Sie nun noch einige Zeit auf der Autobahn. Stellen Sie sich vor, sie hätten während der Fahrt die Möglichkeit mit Ihrem Fahrzeug-Computer einen Dialog wie diesen über gesprochene Sprache zu führen. Ziel des Dialogs wäre es, dass der Computer Ihnen anschließend persönliche, auf Sie individuell zugeschnittene Empfehlungen für Ihren Aufenthalt geben kann.

25. Wie gerne würden Sie in dieser Situation dieses Dialogsystem (in gesprochener Sprache) benutzen?

- außerordentlich gerne
 ziemlich gerne
 mittelmäßig gerne
 eher ungern, Begründung:

gar nicht, Begründung:

26. Den Dialog kann man auch über gesprochene Sprache führen (über Kopfhörer, Mikrofon, Mobiltelefon, etc.). Das hat den Vorteil, dass man dadurch nebenbei einer anderen Tätigkeit nachgehen kann, wie z.B. Auto fahren, spazieren gehen, Hausarbeit (z.B. bügeln), etc. Wären Sie eher bereit, dem System Ihre Interessen ausführlich mitzuteilen, wenn Sie nebenbei einer anderen Tätigkeit nachgehen könnten (als wenn Sie sich dazu an den Computer setzen müssten)?

- außerordentlich ziemlich mittelmäßig kaum gar nicht

27. Wäre Ihre Bereitschaft, diese Art von Dialog auf einer längeren Autobahnfahrt zu führen höher oder geringer als wenn Sie sich dazu zuhause an den Computer setzen müssten?

- höher genau gleich geringer

28. Können Sie sich weitere Alltagssituationen vorstellen, die Ihrer Meinung nach geeignet wären, um einen solchen Dialog (über Tastatur oder gesprochene Sprache) zu führen? Welche?

29. Wenn der Dialog zuhause an Ihrem Computer stattfinden würde, in welcher Form würden Sie ihn dann lieber durchführen:

- gesprochene Spracheingabe und gesprochene Sprachausgabe
 Tastatureingabe und Ausgabe als Text am Bildschirm

5 Interesse und Empfindung

30. Fanden Sie es interessant sich mit dem System zu unterhalten?

- außerordentlich ziemlich mittelmäßig kaum gar nicht

31. Mit welchen der folgenden Sätze würden Sie Ihre Empfindungen während des Dialogs am ehesten beschreiben (mehrere Fälle können angekreuzt werden)?

(0 = trifft gar nicht zu, 5 = trifft voll zu)

- | | | | | | | |
|-------------------------------------|----------------------------|----------------------------|----------------------------|----------------------------|----------------------------|----------------------------|
| 1) Der Dialog hat mir Spaß gemacht. | <input type="checkbox"/> 0 | <input type="checkbox"/> 1 | <input type="checkbox"/> 2 | <input type="checkbox"/> 3 | <input type="checkbox"/> 4 | <input type="checkbox"/> 5 |
| 2) Ich fand den Dialog spannend. | <input type="checkbox"/> 0 | <input type="checkbox"/> 1 | <input type="checkbox"/> 2 | <input type="checkbox"/> 3 | <input type="checkbox"/> 4 | <input type="checkbox"/> 5 |
| 3) Ich habe mich gelangweilt. | <input type="checkbox"/> 0 | <input type="checkbox"/> 1 | <input type="checkbox"/> 2 | <input type="checkbox"/> 3 | <input type="checkbox"/> 4 | <input type="checkbox"/> 5 |
| 4) Ich habe mich geärgert. | <input type="checkbox"/> 0 | <input type="checkbox"/> 1 | <input type="checkbox"/> 2 | <input type="checkbox"/> 3 | <input type="checkbox"/> 4 | <input type="checkbox"/> 5 |
| 5) Sonstiges: | <input type="checkbox"/> 0 | <input type="checkbox"/> 1 | <input type="checkbox"/> 2 | <input type="checkbox"/> 3 | <input type="checkbox"/> 4 | <input type="checkbox"/> 5 |

6 Vorschläge zur Verbesserung

32. Was gefiel Ihnen an diesem Dialog?

33. Hatten Sie das Gefühl, dass das System irgendwelche Fehler gemacht hat?

- sehr häufig häufig gelegentlich selten sehr selten

34. Falls ja, waren diese Fehler störend?

außerordentlich ziemlich mittelmäßig kaum gar nicht

35. Fällt Ihnen etwas ein, wodurch man den Dialog verbessern könnte bzw. welche Funktionen würden Sie sich bei diesem Dialog zusätzlich wünschen?

7 Angaben zur Person

Wie oft fahren Sie selbst oder als Beifahrer/in mit dem Auto?

- täglich
 einige Male in der Woche
 einige Male im Monat
 einige Male im Jahr
 ich fahre gar nicht mit dem Auto (weder als Fahrer noch als Beifahrer)

Haben Sie schon einmal ein anderes Dialogsystem verwendet (z.B. indem Sie bei einer Telefon-Hotline angerufen haben, bei der eine Computerstimme antwortet, oder indem Sie sich mit einem virtuellen/er Assistenten/in auf einer Internetseite unterhalten haben)?

sehr häufig häufig gelegentlich selten sehr selten noch nie

Wie würden Sie Ihre Fähigkeiten bzw. Erfahrung im Umgang mit Computern allgemein beschreiben:

- außerordentlich geübt
 ziemlich geübt
 durchschnittlich geübt
 kaum geübt
 gar nicht geübt

Alter:

Geschlecht:

Ausbildung:

Derzeit ausgeübter Beruf:

Vielen Dank, dass Sie sich die Zeit genommen haben, an dieser Studie teilzunehmen. Sie tragen dadurch ganz wesentlich dazu bei, Dialogsysteme weiter zu erforschen und noch benutzerfreundlicher zu gestalten.

B.2 User Guidelines and Instructions

This appendix contains a cutout of the most important instructions and guidelines we gave to the subjects of our final system evaluation. They have been introduced in Section 6.3.1.3.

Benutzungshinweise für das LINGUINI System

Was ist LINGUINI?

Das System LINGUINI („LINGUistic INterest Identification“) wird sich mit Ihnen in natürlicher Sprache unterhalten, um Ihre persönlichen Interessen kennen zu lernen. Es soll während des Gesprächs versuchen, auf Ihre speziellen Interessen möglichst gut einzugehen und sinnvolle Fragen dazu zu stellen. Sinn des Ganzen ist es, durch den Dialog ein individuelles Interessensprofil für Sie zu erstellen, das später einmal dazu benutzt werden kann, auf speziell für Sie interessante Veranstaltungen und Freizeitangebote hinzuweisen.

Was soll in dieser Studie untersucht werden und was nicht?

Mit der Studie sollen nicht Ihre Interessen beurteilt werden, sondern nur die Fähigkeiten des Systems, diese zu korrekt zu erkennen, damit sinnvoll umzugehen und sich im Gespräch an Sie anzupassen.

I. Hinweise für Ihre Eingaben

Was Sie bitte unbedingt beachten sollten, damit LINGUINI Sie versteht:

1. Achten Sie auf korrekte Groß-Kleinschreibung:

- Hauptwörter („Zeitung“) und Eigennamen („Harry Potter“) immer großschreiben
- Verben („schwimmen“) immer klein schreiben
- zusammengesetzte Verben („saubermachen“; „radfahren“; „ausgehen“) zusammenschreiben (ohne Leerzeichen)
- am Zeilenanfang nicht groß schreiben, es sein denn das Wort ist ein Hauptwort oder Eigenname

2. Vermeiden Sie soweit möglich fremdsprachige Ausdrücke:

Beispiele:

- lieber „Spiele“ statt „Games“
- lieber „Java programmieren“ statt „Java Programming“
- Fremdsprachige Eigennamen wie „Star Wars“, „Badminton“ etc. sind natürlich erlaubt.

3. Verzichten Sie auf Redewendungen:

Redewendungen und Begriffe im übertragenen Sinn können Computersysteme oft nur wörtlich interpretieren und würden sie deshalb falsch verstehen.

- Statt „auf den Putz hauen“ also lieber „Parties“ oder „Nächte durchfeiern“ eingeben.

4. Bitte keine umgangssprachlichen Spezialausdrücke/Symbole (aus z.B. Chat, SMS) eingeben:

- Smilies ☺
- „gesternte“ Kommentare (*LOL*, *g*, etc.)
- Einschübe („ähm“, „well“, „ach“, „okay“, etc.).

5. Achten Sie darauf, Tippfehler zu vermeiden:

Im derzeitigen Stadium kann das System noch nicht damit umgehen und würde Sie daher nicht verstehen.

6. Falls das System Ihnen einige Antworten zur Auswahl gibt:

Falls das System Ihnen mehrere Antwortmöglichkeiten präsentiert, wählen Sie bitte eine aus und geben Sie diese in genau der selben Schreibweise ein.

Der Einfachheit halber geben Sie bei ja-nein-Fragen bitte immer nur "ja" oder "nein" ein, auch wenn Sie nur leicht in die jeweilige Richtung tendieren.

7. Zuletzt einige Beispiele für sinnvolle Eingaben. Sie dürfen sowohl Interessen als auch Abneigungen eingeben:

öfters mit Freunden essen gehen
guter spanischer Wein
besonders Harry Potter
Trivial Pursuit spielen
im Herbst tauchen im Roten Meer
alleine tanzen
sehr gerne wandern
Bücher

keinesfalls tauchen in Australien
keine Krimis von Agatha Christie
nicht radfahren
deutsche Schlager auf keinen Fall
ins Theater gehen niemals
Sprachen lernen nicht so sehr

Sie können beliebig viele Interessen/Abneigungen auf einmal eingeben, indem Sie sie durch Semikolon „;“ trennen:

öfters mit Freunden essen gehen; guter spanischer Wein; keinesfalls wandern

III Anleitung zur Benutzung

Gehen Sie bitte folgendermaßen vor:

1) Stellen Sie sich folgende Anwendungssituation vor:

LINGUINI soll später im Auto durch gesprochene Sprache zum Einsatz kommen. In dieser Benutzerstudie wird der Dialog jedoch über Tastatur am Computer geführt. Versuchen Sie bitte, sich vor dem Dialog deshalb in diese Situation möglichst gut hineinzusetzen:

"Sie fahren im Auto (als Fahrer/in oder Beifahrer/in). Sie beschließen spontan einen Abstecher in eine nahe gelegene größere Stadt zu machen. Leider hatten Sie vor der Reise keine Zeit, sich über Freizeitmöglichkeiten, Veranstaltungen und Attraktionen in dieser Stadt zu informieren. Um die Stadt zu erreichen fahren Sie nun noch einige Zeit auf der Autobahn. Stellen Sie sich vor, Sie hätten während der Fahrt die Möglichkeit mit Ihrem Fahrzeug-Computer einen Dialog über gesprochene Sprache zu führen. Ziel des Dialogs wäre es, dass der Computer Ihnen anschließend persönliche, auf Sie individuell zugeschnittene Empfehlungen für Ihren Aufenthalt geben kann."

2) Führen Sie jetzt den Dialog mit LINGUINI:

1. Internet-Browser starten und bitte zunächst sicherstellen, dass Cookies zugelassen sind:

Falls Sie Mozilla-Firefox verwenden: Im Menü öffnen: „Extras“ → „Einstellungen“ → links „Datenschutz“ auswählen → rechts: das Feld „Cookies akzeptieren“ muss ein Kreuzchen enthalten, falls es das nicht tut, bitte anklicken → rechts unten „ok“ klicken.

Falls Sie Microsoft Internet-Explorer verwenden: Falls Sie die Standardeinstellungen verwenden (also selbst nichts verändert haben) müssen Sie nichts ändern. Ansonsten: Extras → Internetoptionen → Datenschutz → Der Regler sollte in der Mitte auf „identifizierbare Informationen ohne stillschweigende Zustimmung verwenden“ stehen → unten auf „ok“ klicken.

2. Jetzt bitte im Browser folgende Internet-Adresse/URL eingeben:

<http://atschlichter38.informatik.tu-muenchen.de:8011/servlet/CustomerDialog?cmd=info>

3. Sie sehen ein Fenster zur Anmeldung. Geben Sie hier bitte als Login „dialogtest“ und als Password „test“ ein. Klicken Sie dann auf "Login using Cookies".

4. Nun links oben auf „Dialog-System“ klicken. Jetzt sehen Sie jetzt die derzeitige Benutzeroberfläche von LINGUINI.

Wie wird der Dialog ablaufen:

LINGUINI stellt Ihnen im grauen Fenster eine Frage, auf die Sie bitte im weißen Eingabefeld eine Antwort eintippen und dann auf "send answer" klicken. LINGUINI stellt dann die nächste Frage usw. Versuchen Sie im Dialog bitte jede Frage zu beantworten.

LINGUINI kann übrigens **noch keine korrekten Wortendungen** erzeugen. Die Fragen, die es stellt, klingen daher manchmal etwas ungrammatikalisch.

Wann endet der Dialog:

LINGUINI sagt bescheid, wenn es genügend Informationen bekommen hat, der Dialog endet dann. Sollten Sie selbst den Dialog vorzeitig beenden wollen, geben Sie bitte „ende“ ein und klicken Sie auf „send answer“.

5. **WICHTIG:** Notieren Sie bitte unbedingt die **Profil-ID** (Nummer Ihres Profils), die rechts oben angezeigt wird, sobald sie erscheint.

6. Wenn der Dialog zu Ende ist, können Sie sich den gesamten **Dialogverlauf** nochmals ansehen in dem grauen Fenster unterhalb Ihres Eingabefeldes. Das kann hilfreich sein beim Beantworten des Fragebogens. Lassen Sie den Browser also am besten weiter geöffnet, während Sie den Fragebogen ausfüllen. **Achtung:** Klicken Sie beim Betrachten des Dialogverlaufs nicht auf „Seite neu laden“ in Ihrem Browser!

3) Füllen Sie jetzt bitte den Fragebogen aus:

Speichern Sie den Fragebogen (fragebogen.rtf) auf Ihrem Computer.

Sie können nun zwischen 2 Möglichkeiten wählen:

- Fragebogen mit Microsoft Word öffnen, ausdrucken und von Hand ausfüllen; per Fax oder Briefpost zurücksenden
- Fragebogen mit Microsoft Word öffnen und direkt am Computer ausfüllen → dabei bitte "x" zum ankreuzen der []-Kästchen verwenden und in die grauen Felder Freitext einfach eintippen; per Email zurücksenden

4) Bevor Sie Ihren Internet-Browser schließen, loggen Sie sich bitte unbedingt aus dem Dialogsystem aus: Dazu links oben auf „Logout“ klicken.

B.3 Sample Dialogs

The following original dialogs conducted by three different users are taken from our final user study (cf. Section 6.3). The first two protocols are rather positive examples for dialogs that can be conducted with LINGUINI, whereas the third dialog has been less successful.

1) Dialog and Profile ID: 42039

Starteingabe: Wein trinken; Freunde treffen; Brettspiele spielen

Frage (1): Was interessiert Sie sonst noch?

Antwort: radfahren; Musik hören; Zeitschriften lesen

Frage (2): A propos Zeitschriften lesen : Mögen Sie Sportzeitschrift ? (ja, nein)

Antwort: nein

Frage (3): Welche/s/n Lieblings- Zeitschriften haben Sie?

Antwort: The New Yorker

Frage (4): radfahren Sie Radfahren ? (ja, nein)

Antwort: ja

Frage (5): Sie sagten, Sie hören Musik . Interessieren Sie sich auch für Musik machen ? (ja, nein)

Antwort: nein

Frage (6): Beschreiben Sie bitte 2-3 Interessen, denen Sie vorwiegend außer Haus nachgehen!

Antwort: Spazieren gehen; im See schwimmen; Tiere beobachten

Frage (7): Wie wichtig ist Ihnen Spazieren gehen ? (sehr_wichtig, wichtig, weniger_wichtig)

Antwort: wichtig

Frage (8): Da Sie sich für Tiere interessieren, interessiert Sie auch das Thema Pferde ? (ja, nein)

Antwort: nein

Frage (9): Wie häufig gehen Sie Spazieren ? (regelmäßig, häufig, gelegentlich, selten)

Antwort: gelegentlich

Frage (10): Mit welchem Ort steht für Sie Spazieren gehen in Verbindung?

Antwort: Park

Frage (11): Interessieren Sie sich für Freunde treffen ? (ja, nein)

Antwort: ja

Frage (12): Da Sie Wein erwähnt haben, interessieren Sie sich auch für Essen allgemein? (ja, nein)

Antwort: ja

Frage (13): Da Sie sich für Essen interessieren, interessiert Sie auch das Thema Gerichte ? (ja, nein)

Antwort: ja

Frage (14): Da Sie Spazieren erwähnt haben, interessieren Sie sich auch für Sport allgemein? (ja, nein)

Antwort: nein

Frage (15): Da Sie sich für Tiere interessieren, interessiert Sie auch das Thema Hunde ? (ja, nein)

Antwort: nein

Frage (16): Da Sie Essen erwähnt haben, interessieren Sie sich auch für Stadt allgemein? (ja, nein)

Antwort: ja

Frage (17): Wie wichtig ist Ihnen Stadt ? (sehr_wichtig, wichtig, weniger_wichtig)
Antwort: sehr_wichtig

Frage (18): Welche Aktivität (=VERB) verbinden Sie mit (dem Interesse) Stadt ?
Antwort: schlendern

Frage (19): Mit welcher anderen Person gehen Sie dem Interesse Stadt schlendern nach? (Ansonsten geben Sie bitte 'alleine' ein.)
Antwort: mit meiner Freundin

Frage (20): Bitte geben Sie noch ein paar Interessen ein!
Antwort: Essen kochen; reisen; im Internet surfen

Frage (21): Beschreiben Sie bitte 2-3 Interessen, denen Sie vorwiegend zuhause nachgehen!
Antwort: Fernsehserien schauen; telefonieren

Frage (22): Welche Dinge, Themen oder Aktivitäten interessieren Sie noch?
Antwort: Auto fahren; tibetische Medizin; Radio hören

Frage (23): Welche/s/n Lieblings- Auto (= NOMEN) haben Sie?
Antwort: Twingo

Frage (24): Wie häufig fahren Sie Auto ? (regelmäßig, häufig, gelegentlich, selten)
Antwort: häufig

Frage (25): Mit welchem Ort steht für Sie Auto fahren in Verbindung?
Antwort: München

Frage (26): Bitte schreiben Sie jetzt die Profil-ID, die rechts oben am Bildschirm angezeigt wird, auf Ihren Fragebogen. Vielen Dank für Ihre Zeit. Der Dialog endet jetzt. Die Informationen wurden Ihrem Benutzerprofil hinzugefügt.

2) Dialog and Profile ID: 42182

Starteingabe: Malerei; Musik hören; lesen

Frage (1): Da Sie Malerei erwähnt haben, interessieren Sie sich auch für Kunst allgemein? (ja, nein)
Antwort: ja

Frage (2): Da Sie sich für Kunst interessieren, interessiert Sie auch das Thema Schlager ? (ja, nein)
Antwort: nein

Frage (3): Da Sie sich für Kunst interessieren, interessiert Sie auch das Thema Musik ? (ja, nein)
Antwort: ja

Frage (4): machen Sie Musik ? (ja, nein)
Antwort: nein

Frage (5): Welche Aktivität (= VERB) verbinden Sie (dann) mit (dem Interesse) Musik ?
Antwort: hören;tanzen

Frage (6): Mit welcher anderen Person gehen Sie dem Interesse Kunst nach? (Ansonsten geben Sie bitte 'alleine' ein.)
Antwort: alleine

Frage (7): Welche Aktivität (=VERB) verbinden Sie mit (dem Interesse) Kunst ?
Antwort: sehen;malen;hören;fühlen;denken

Frage (8): Mit welchem Ort steht für Sie Kunst sehen,malen,hören,fühlen,denken in Verbindung?
Antwort: Natur;Atelier;Museum;

Frage (9): Bitte geben Sie noch ein paar Interessen ein!

Antwort: Büchern;tanzen;reisen

Frage (10): Mit welchem Ort steht für Sie Büchern in Verbindung?

Antwort: Bibliothek;Buchhandlung,Lesung

Frage (11): Welche Aktivität (=VERB) verbinden Sie mit (dem Interesse) Büchern ?

Antwort: lesen; zuhören

Frage (12): Welche/s/n Lieblings- Büchern (= NOMEN) haben Sie?

Antwort: Romane;Sachbücher Psychologie

Frage (13): Interessieren Sie sich für Radfahren ? (ja, nein)

Antwort: ja

Frage (14): Welche Aktivität (=VERB) verbinden Sie mit (dem Interesse) Radfahren ?

Antwort: bewegen

Frage (15): Mit welcher anderen Person gehen Sie dem Interesse Radfahren bewegen nach? (Ansonsten geben Sie bitte 'alleine' ein.)

Antwort: alleine

Frage (16): Wie wichtig ist Ihnen Radfahren bewegen ? (sehr_wichtig, wichtig, weniger_wichtig)

Antwort: wichtig

Frage (17): Interessieren Sie sich für Sendungen ? (ja, nein)

Antwort: ja

Frage (18): Da Sie sich für Sendungen interessieren, interessiert Sie auch das Thema Quizshow ? (ja, nein)

Antwort: nein

Frage (19): A propos Sendungen : Mögen Sie Kochsendungen ? (ja, nein)

Antwort: ja

Frage (20): Mit welcher anderen Person gehen Sie dem Interesse Sendungen nach? (Ansonsten geben Sie bitte 'alleine' ein.)

Antwort: alleine

Frage (21): Welche Aktivität (=VERB) verbinden Sie mit (dem Interesse) Sendungen ?

Antwort: zuschauen, zuhören

Frage (22): ansehen Sie Sendungen ? (ja, nein)

Antwort: ja

Frage (23): A propos Sendungen ansehen : Mögen Sie Naturfilm ? (ja, nein)

Antwort: ja

Frage (24): Mit welcher anderen Person gehen Sie dem Interesse Sendungen ansehen nach? (Ansonsten geben Sie bitte 'alleine' ein.)

Antwort: alleine

Frage (25): Beschreiben Sie bitte 2-3 Interessen, denen Sie vorwiegend zuhause nachgehen!

Antwort: kochen, baden

Frage (26): Was interessiert Sie sonst noch?

Antwort: Reden mit Freunden; spaziergehen; Sauna

Frage (27): Wie wichtig ist Ihnen Sauna ? (sehr_wichtig, wichtig, weniger_wichtig)

Antwort: wichtig

Frage (28): Mit welcher anderen Person gehen Sie dem Interesse Sauna nach? (Ansonsten geben Sie bitte 'alleine' ein.)

Antwort: alleine

Frage (29): Welche/s/n Lieblings- Sauna (= NOMEN) haben Sie?

Antwort: Finnische Sauna

Frage (30): Bitte geben Sie noch ein Interesse ein!

Antwort: ausruhen

Frage (31): Bitte schreiben Sie jetzt die Profil-ID, die rechts oben am Bildschirm angezeigt wird, auf Ihren Fragebogen. Vielen Dank für Ihre Zeit. Der Dialog endet jetzt. Die Informationen wurden Ihrem Benutzerprofil hinzugefügt.

3) Dialog and Profile ID: 42157

Starteingabe: lesen;schwimmen;spaziergehen

Frage (1): Was interessiert Sie sonst noch?

Antwort: Architektur

Frage (2): Da Sie Architektur erwähnt haben, interessieren Sie sich auch für Wissenschaft allgemein? (ja, nein)

Antwort: ja

Frage (3): Da Sie sich für Wissenschaft interessieren, interessiert Sie auch das Thema Technik ? (ja, nein)

Antwort: nein

Frage (4): Mit welchem Ort steht für Sie Wissenschaft in Verbindung?

Antwort: Bibliothek

Frage (5): Wie wichtig ist Ihnen Wissenschaft ? (sehr_wichtig, wichtig, weniger_wichtig)

Antwort: wichtig

Frage (6): Welche Aktivität (=VERB) verbinden Sie mit (dem Interesse) Wissenschaft ?

Antwort: anschauen

Frage (7): Bitte geben Sie noch ein paar Interessen ein!

Antwort: malen;musizieren;einrichten

Frage (8): Beschreiben Sie bitte 2-3 Interessen, denen Sie vorwiegend zuhause nachgehen!

Antwort: lesen;renovieren;dekorieren

Frage (9): Beschreiben Sie bitte 2-3 Interessen, denen Sie vorwiegend außer Haus nachgehen!

Antwort: radfahren;spaziergehen;gärtnern

Frage (10): radfahren Sie Radfahren ? (ja, nein)

Antwort: ja

Frage (11): Welche Dinge, Themen oder Aktivitäten interessieren Sie noch?

Antwort: Schwimmbäder

Frage (12): Da Sie sich für Schwimmbäder interessieren, interessiert Sie auch das Thema Sauna ? (ja, nein)

Antwort: nein

Frage (13): Interessieren Sie sich für Radfahren ? (ja, nein)

Antwort: ja

Frage (14): Bitte geben Sie noch ein Interesse ein!

Antwort: Gartenbau

Frage (15): Interessieren Sie sich für Wellness ? (ja, nein)

Antwort: ja

Frage (16): Wie wichtig ist Ihnen Wellness ? (sehr_wichtig, wichtig, weniger_wichtig)

Antwort: weniger_wichtig

Frage (17): Welche Aktivität (=VERB) verbinden Sie mit (dem Interesse) Wellness ?

Antwort: ausruhen

Frage (18): Welche/s/n Lieblings- Wellness (=NOMEN) haben Sie?

Antwort: keines

Frage (19): Bitte schreiben Sie jetzt die Profil-ID, die rechts oben am Bildschirm angezeigt wird, auf Ihren Fragebogen. Vielen Dank für Ihre Zeit. Der Dialog endet jetzt. Die Informationen wurden Ihrem Benutzerprofil hinzugefügt.

B.4 The Treatment of Outlier Values

In Section 6.3.1.4, we mentioned that one of the questionnaires (Questionnaire 42187) returned by the users could not be used for evaluation because the majority of its quantitative values were outlier values (or: outliers), i.e. they were not representative for the group of test users.

Figure B.1 shows the quantitative questions (F1, ..., F34) in relation to the possible answer values (0 – 5). Qualitative questions, in which users explained their impressions or ideas freely, have been omitted in this figure, because they cannot directly be compared with quantitative values.

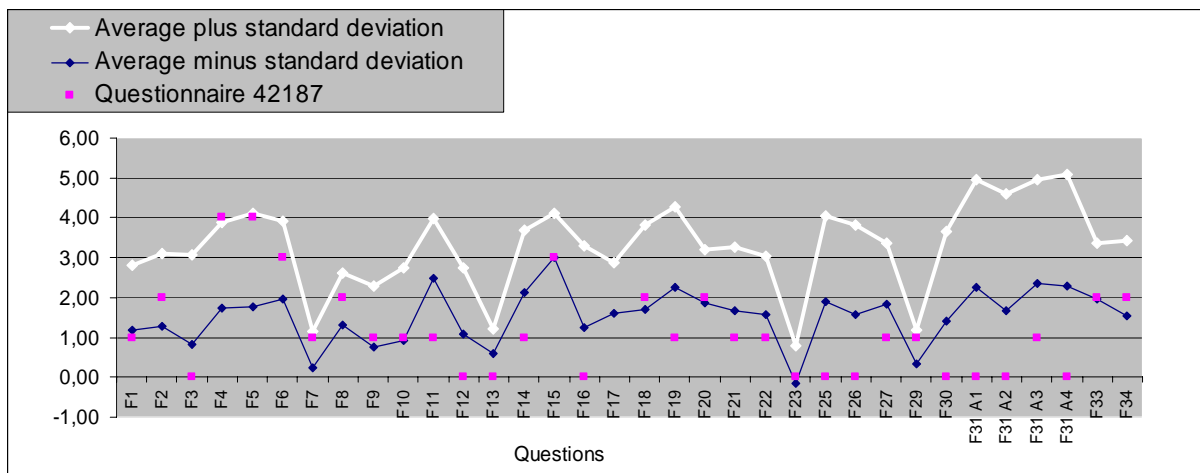


Figure B.1: Outlier values

First, we calculated the standard deviation and the average value for each question. In the figure shown above, the upper curve represents the **average answer value** of all other users for each question **plus the standard deviation**. The lower curve represents the **average value minus the standard deviation**. The particular answer values of Questionnaire 42187 are indicated in the figure as individual dots.

For 29 questionnaires, the **majority of the answer values** always lies *between* these two curves, i.e. within the standard deviation. Only for Questionnaire 42187, the majority of the answer values lies *outside both curves*, i.e. outside the standard deviation. For this reason, Questionnaire 42187 cannot be considered representative for the group of test users and has therefore been eliminated from the data set.

This procedure is supported by established evaluation methods as, for example, described in [Bortz and Döring 2002]. According to the authors, it has to be checked carefully that **aggregated values** (i.e. values that comprise several values of individual users) calculated in an evaluation in fact adequately represent the feature distribution within the group of test users. Therefore, before calculating aggregated values, [Bortz and Döring 2002] recommend first gaining an overview of the data set by visualizing it graphically (which we did by means of the standard deviation figure). Then, outliers have to be identified and eliminated from the data set in order not to adulterate aggregated values, such as the **average value** for example. According to [Bortz and Döring 2002], one of the most typical errors made in evaluations is not to notice and exclude outliers in this context.

Bibliography

- [Abbattista et al. 2002a] F. Abbattista, M. Degemmis, N. Fanizzi, O. Licchelli, P. Lops, G. Semeraro, and F. Zambetta. Learning User Profiles for Content-Based Filtering in E-Commerce. In *Proceedings AI*AI Workshop su Apprendimento Automatico: Metodi e Applicazioni*, Sienna, Italy, 2002.
- [Abbattista et al. 2002b] F. Abbattista, M. Degemmis, O. Licchelli, P. Lops, G. Semeraro, and F. Zambetta. Improving the Usability of an E-Commerce Web Site through Personalization. In F. Ricci and B. Smyth (eds). *Recommendation and Personalization in eCommerce, Proc. of the Workshop on Recommendation and Personalization in Electronic Commerce, 2nd Int. Conf. on Adaptive Hypermedia and Adaptive Web Based Systems*, pp. 20–29, Malaga, Spain, May, 2002.
- [Abney 1996a] S. Abney. Part-of-Speech Tagging and Partial Parsing. In K. Church, S. Young, and G. Bloothoof (eds). *Corpus-Based Methods in Language and Speech*. Dordrecht, Kluwer Academic Publishers, 1996.
- [Abney 1996b] S. Abney. Partial Parsing via Finite-State Cascades. In *Proceedings of the ESSLLI '96 Robust Parsing Workshop*, 1996.
- [Akoulchina and Ganascia 1997] I. Akoulchina and J.-G. Ganascia. SATELIT-Agent: an adaptive interface based on learning interface agents technology. In A. Jameson, C. Paris, and C. Tasso (eds). *User Modeling: Proceedings of the Sixth International Conference*, Springer, pp. 21–32, 1997.
- [Alheit et al. 1994] P. Alheit, S. Völker, B. Westermann, and M. M. Zwick. Die Kehrseite der “Erlebnisgesellschaft” – Eine explorative Studie. In P. Alheit and A. Keil (eds). *Werkstattberichte des Instituts für angewandte Biographie- und Lebensweltforschung [IBL]*. Band 1, Universität Bremen, Universitätsbuchhandlung Bremen (ISBN: 3-88722-325-X), 1994.
- [Allen et al. 2001] J. Allen, G. Ferguson, and A. Stent. An architecture for more realistic conversational systems. In *IUI '01: Proceedings of the 6th international conference on intelligent user interfaces*, Santa Fe, New Mexico, USA, pp. 1–8, ACM Press, 2001.
- [Appelt 1985] D.E. Appelt. *Planning English Sentences*. Cambridge University Press, 1985.
- [Ardissono 1996] L. Ardissono. *Dynamic User Modeling and Plan Recognition in Dialogue*. PhD Thesis, Dipartimento di Informatica, Università di Torino, Italy, 1996.
- [Ardissono et al. 2001] L. Ardissono, A. Felfernig, G. Friedrich, D. Jannach, M. Zanker, and R. Schäfer. Intelligent interfaces for distributed web-based product and service configu-

- ration. In *Proceeding of the First Asia-Pacific Conference on Web Intelligence (WI-2001)*, Maebashi City, Japan, 2001.
- [Ardissono and Goy 2000] L. Ardissono and A. Goy. Tailoring the Interaction with Users in Web Stores. *User Modeling and User-Adapted Interaction*, 10(4): 251–303, Apr. 2000.
- [Ardissono et al. 1999] L. Ardissono, A. Goy, R. Meo, G. Petrone, L. Console, L. Lesmo, C. Simone, and P. Torasso. A Configurable System for the Construction of Adaptive Virtual Stores. *World Wide Web*, 2(3): 143–159, 1999.
- [Ardissono et al. 2003] L. Ardissono, A. Goy, G. Petrone, M. Segnan, and P. Torasso. INTRIGUE: Personalized recommendation of tourist attractions for desktop and handset devices. *Applied Artificial Intelligence: Special Issue on Artificial Intelligence for Cultural Heritage and Digital Libraries*, 17(8-9): 687–714, 2003.
- [Asadi et al. 1989] A. Asadi, R. Schwartz, and J. Makhoul. Automatic Detection of New Words in a Large Vocabulary Continuous Speech Recognition System. In *Proc. of the Speech and Natural Language Workshop*, Cape Cod, MA, pp. 263-265, 1989.
- [Ast 2003] M. Ast. *Entwurf und Implementierung einer Java-API für GermaNet*. Systementwicklungsprojekt, Technische Universität München, Nov. 2003.
- [Aust et al. 1995] H. Aust, M. Oerder, F. Seide, and V. Steinbiss. The Philips automatic train timetable information system. *Speech Commun.*, 17(3-4): 249-262, Nov. 1995.
- [Balbanovic and Shoham 1997] M. Balbanovic and Y. Shoham. Fab – Content-Based, Collaborative Recommendation. *Communications of the ACM*, 40(3): 66-72, 1997.
- [Bateman 1997] J. Bateman. Deep Generation. In R. A. Cole, J. Mariani, H. Uszkoreit, G. Varile, A. Zaenen, V. Zue, and A. Zampolli (eds). *Survey of the State of the Art in Human Language Technology, Studies in Natural Language Processing, Linguistica Computazionale*, Vol. XII-XIII, Cambridge University Press/Giardini Editori E Stampatori, Pisa, 1997.
- [Bauer 2004] M. Bauer. Transparent User Modeling for a Mobile Personal Assistant. In *Working Notes of the Annual Workshop of the SIG on Adaptivity and user Modeling in Interactive Software Systems of the GI*, pp. 3–8, 2004.
- [Bering et al. 2003] C. Bering, W. Drozdzyński, G. Erbach, C. Guasch, P. Homola, S. Lehmann, L. Hong, H.-U. Krieger, J. Piskorski, U. Schäfer, A. Shimada, M. Siegel, F. Xu, and D. Ziegler-Eisele. Corpora and evaluation tools for multilingual named entity grammar development. In *Proceedings of Multilingual Corpora Workshop at Corpus Linguistics 2003*, pp. 42-52, Lancaster, 2003.
- [Berthold and Jameson 1999] A. Berthold and A. Jameson. Interpreting Symptoms of Cognitive Load in Speech Input. In *UM99 – Proceedings of the Seventh International Conference on User Modeling*, pp. 235–144, Banff, Canada, 1999.
- [Billsus et al. 2002] D. Billsus, C. A. Brunk, C. Evans, B. Gladish, and M. Pazzani. Adaptive Interfaces for Ubiquitous Web Access. *Commun. of the ACM*, 45(5): 34-38, May 2002.

- [Billsus and Pazzani 1999] D. Billsus and M. Pazzani. A Hybrid User Model for News Story Classification. In *UM 99 – Proc. of the Seventh International Conference on User Modeling*, Banff, Canada, 1999.
- [Bortz and Döring 2002] J. Bortz and N. Döring. *Forschungsmethoden und Evaluation für Human- und Sozialwissenschaftler*. Springer, 3. überarb. Auflage, 2002.
- [Brants 2000] T. Brants. TnT - A Statistical Part-of-Speech Tagger. In *Proceedings of the Sixth Applied Natural Language Processing Conference ANLP-2000*, Seattle, WA, 2000.
- [Brants et al. 1997] T. Brants, R. Hendriks, S. Kramp, B. Krenn, C. Preis, W. Skut, and H. Uszkoreit. *Das NEGRA-Annotationsschema*. NEGRA Project Report, Universität des Saarlandes, Saarbrücken, Germany, 1997.
- [Breese et al. 1998] J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, Wisconsin, USA, 1998.
- [Bresnan 2001] J. Bresnan. *Lexical Functional Syntax*. Blackwell, 2001.
- [Brill 1995] E. Brill. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part of Speech Tagging. *Computational Linguistics* 21(4): 543-565, Dec. 1995.
- [Brockmann 2002] C. Brockmann. *Evaluating and combining approaches to selectional preference acquisition*. Master's Thesis, Universität des Saarlandes, Saarbrücken, Germany, 2002.
- [Brusilovsky 1996] P. Brusilovsky. Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction* 6 (2-3): 87-129, 1996.
- [Brusilovsky and Maybury 2002] P. Brusilovsky and M. Maybury. From Adaptive Hypermedia to the Adaptive Web. *Communications of the ACM* 45(5): 31-33, 2002.
- [Budanitsky and Hirst 2001] A. Budanitsky and G. Hirst. Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures. *Workshop on WordNet and Other Lexical Resources, Second Meeting of the North American Chapter of the Association for Computational Linguistics*, Pittsburgh, USA, June 2001.
- [Bühler et al. 2002] D. Bühler, W. Minker, J. Häussler, and S. Krüger. Flexible Multimodal Human-Machine Interaction in Mobile Environments. *ECAI Workshop on Artificial Intelligence in Mobile System (AIMS)*, Lyon, France, 2002.
- [Bühler et al. 2003] D. Bühler, S. Vignier, P. Heisterkamp, and W. Minker. Safety and operating issues for mobile human-machine interfaces. *IUI '03 – Proceedings of the 8th international conference on intelligent user interfaces*, pp. 227—229, Miami, Florida, USA, 2003.
- [Burke 2002] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4): 331-370, 2002.

- [Busemann 1996] S. Busemann. Best-First Surface Realization. In D. Scott (ed). *Proceedings of the 8th International Natural Language Generation Workshop (INLG'96)*, pp. 101-110, Sussex, UK, June 1996.
- [Busemann 2003] S. Busemann. Generierung natürlichsprachlicher Texte. In G. Görz, C.-R. Rollinger, and J. Schneeberger (eds). *Handbuch der Künstlichen Intelligenz*, Oldenbourg, 2003.
- [Byers et al. 2004] S. Byers, L. Cranor, D. Kormann, and P. McDaniel. Searching for Privacy: Design and Implementation of a P3P-Enabled Search Engine. In *Proc. of the 2004 Workshop on Privacy Enhancing Technologies (PET2004)*, Toronto, Canada, 2004.
- [Carstensen et al. 2004] K.-U. Carstensen, C. Ebert, C. Endriss, S. Jekat, R. Klabunde, and H. Langer (eds). *Computerlinguistik und Sprachtechnologie. Eine Einführung*. Heidelberg, Berlin: Spektrum Akademischer Verlag, 2004.
- [Chai et al. 2001] J. Chai, V. Horvath, N. Nicolov, M. Stys-Budzikowska, N. Kambhatala, and W. Zadrozny. Natural Language Sales Assistant - A Web-based Dialog System for Online Sales. In *Proc. of the Thirteenth Innovative Application of Artificial Intelligence Conference (IAAI2001)*, Seattle, USA, pp. 19-26, 2001.
- [Charniak 1997] E. Charniak. Statistical Techniques for Natural Language Parsing. *AI Magazine* 18(4): pp 33-44, 1997.
- [Cheverest et al. 2000] K. Cheverest, N. Davies, K. Mitchell, A. Friday, and C. Efstratiou. Developing a Context-Aware Electronic Tourist Guide: Some Issues and Experiences. In *Proc. Human factors in computing systems, CHI'2000*, The Hague, The Netherlands, pp. 17-24, 2000.
- [Cranor 2003] L. F. Cranor. I Didn't Buy it for Myself: Privacy and Ecommerce Personalization. In *Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society*, Washington, DC, Oct. 2003.
- [Curran and Clark 2003] J. R. Curran and S. Clark. Language independent NER using a maximum entropy tagger. In *Proceedings of the Seventh Conference on Natural Language Learning (CoNLL-03)*, pp. 164-167, Edmonton, Canada, 2003.
- [De Mori and Brugnara 1997] R. De Mori and F. Brugnara. HMM Methods in Speech Recognition. In R. A. Cole, J. Mariani, H. Uszkoreit, G. Varile, A. Zaenen, V. Zue, and A. Zampolli (eds.). *Survey of the State of the Art in Human Language Technology, Studies in Natural Language Processing, Linguistica Computazionale*, Vol. XII-XIII, Cambridge University Press/Giardini Editori E Stampatori, Pisa, 1997.
- [Denaux et al. 2005] R. Denaux, V. Dimitrova, and L. Aroyo. Integrating Open User Modeling and Learning Content Management for the Semantic Web. In *Proceedings of the 10th International Conference on User Modeling*, Edinburgh, UK, July 2005.
- [Diaz-Bone 2004] R. Diaz-Bone. Milieumodelle und Milieuinstrumente in der Marktforschung. *Forum: Qualitative Sozialforschung* 5(2), Art. 28, Mai 2004.

- [Eiseler 2005] T. Eiseler. *Weiterentwicklung eines Dialog Managers für ein Profilerhebungssystem: Auflösung von Mehrdeutigkeiten im benutzeradaptiven, natürlichsprachlichen Dialog mit dem Kunden*. Diploma Thesis, Technische Universität München, Nov. 2005.
- [Felfernig et al. 2004] A. Felfernig, G. Fliedl, and C. Kop. Applying Natural-Language Processing to Knowledge-Based Configuration. In *Proceedings of the Workshop on Configuration, 16th European Conference on Artificial Intelligence (ECAI-2004)*, Valencia, Spain, 2004.
- [Ferguson et al. 1996] G. Ferguson, J. F. Allen, B. W. Miller, and Eric K. Ringger. The design and implementation of the TRAINS-96 system: A prototype mixed-initiative planning assistant. *TRAINS Technical Note 96-5*, Department of Computer Science, University of Rochester, 1996.
- [Finin 1989] T. W. Finin. GUMS: A general user modeling shell. In A. Kobsa and W. Wahlster (eds). *User Models in Dialog Systems*. Berlin, Heidelberg: Springer, pp. 411-430, 1989.
- [Finin and Drager 1986] T. W. Finin and D. Drager. A general user modeling system. In *Proc. of the 6th Canadian Conference on Artificial Intelligence*, Montreal, Canada, pp. 24-29, 1986.
- [Fink and Kobsa 2000] F. Fink and A. Kobsa. A review and analysis of commercial user modeling servers for personalization on the WWW. *User Modeling and User-Adapted Interaction* 10: 209-249, 2000.
- [Fink et al. 1997] J. Fink, A. Kobsa, and I. Jaceniak. Individualisierung von Benutzerschnittstellen mit Hilfe von Datenchips für Personalisierungsinformation. *GMD-Spiegel* 1/1997: 16-17, 1997.
- [Fink et al. 1996] J. Fink, A. Kobsa, and A. Nill. User-Oriented Adaptivity and Adaptability in the AVANTI Project. In *Proceedings of the Conference Designing for the Web: Empirical Studies*, Redmond, WA, Oct. 1996.
- [Fliedner 2002] G. Fliedner. A System for Checking NP Agreement in German Texts. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02), Student Workshop*, pp. 12–17, Philadelphia, PA, July 2002.
- [Furui in 1997] S. Furui. Speaker Recognition. In R. A. Cole, J. Mariani, H. Uszkoreit, G. Varile, A. Zaenen, V. Zue, and A. Zampolli (eds.). *Survey of the State of the Art in Human Language Technology, Studies in Natural Language Processing, Linguistica Computazionale*, Vol. XII-XIII, Cambridge University Press/Giardini Editori E Stampatori, Pisa, 1997.
- [Gärtner 2001] U. Gärtner. The SENECA Project: Speech Recognition within the Car for Entertainment and Communication Systems. *Detroit Auto Interior Show*, 2001.
- [Gärtner et al. 2001] U. Gärtner, W. König, and T. Wittig. Evaluation of manual vs. speech input when using a driver information system in real traffic. *International Driving Sym-*

- posium on Human Factors in Driver Assessment, Training and Vehicle Design*, Aspen, Colorado, 2001.
- [Garside et al. 1997] R. Garside, G. Leech, and A. McEnery. *Corpus Annotation*. Longman, London, New York, 1997.
- [Ghani and Fano 2002] R. Ghani and A. Fano. Building Recommender Systems Using a Knowledge Base of Product Semantics. *Workshop on Recommendation and Personalization in ECommerce at the Second International Conference on Adaptive Hypermedia and Adaptive Web-based Systems*, Malaga, Spain, 2002.
- [Ginsburg 2001] M. Ginsburg. Growing Out of Its Skin: Principles of the Evolution and Extension of the Internet Chess Club, 1995 to Present. In D. Strong and D. Straub (eds). *Proceedings of the Seventh Americas Conference on Information Systems (AMCIS 2001)*, pp. 1514 – 1521, Boston, MA, Omnipress, 2001.
- [Goddeau et al. 1996] D. Goddeau, H. Meng, J. Polifroni, S. Seneff, and S. Busayapongchai. A form-based dialogue manager for spoken language applications. In *Proc. of 4th International Conference on Spoken Language Processing (ICSLP'96)*, pp. 701–704, Pittsburgh, PA, 1996.
- [Göker and Thompson 2000] M. H. Göker and C. A. Thompson. Personalized Conversational Case-Based Recommendation. In *Proceedings of the 5th European Workshop on Advances in Case-Based Reasoning*, E. Blanzieri and L. Portinale (eds). *Lecture Notes In Computer Science*, Vol. 1898: 99-111, London: Springer, 2000.
- [Good et al. 1999] N. Good, J. B. Schafer, J. A. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl. Combining Collaborative Filtering with Personal Agents for Better Recommendations. GroupLens Research Project. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pp. 439-446, USA, 1999.
- [Grewendorf et al. 1987] G. Grewendorf, F. Hamm, and W. Sternefeld. *Sprachliches Wissen. Eine Einführung in moderne Theorien der grammatischen Beschreibung*. Suhrkamp, Frankfurt, 1987.
- [Groh 2005] G. Groh. *Ad-Hoc-Groups in Mobile Communities – Detection, Modeling and Applications*. Doctoral Thesis, Technische Universität München, March, 2005.
- [Gurevych and Niederlich 2005a] I. Gurevych and H. Niederlich. Measuring Semantic Relatedness of GermaNet Concepts. In B. Fisseni, H.-C. Schmitz, B. Schröder, and P. Wagner (eds). *Sprachtechnologie, mobile Kommunikation und linguistische Ressourcen: Applications of GermaNet II*, pp. 462-474, Frankfurt am Main: Peter Lang, 2005.
- [Gurevych and Niederlich 2005b] I. Gurevych and H. Niederlich. Accessing GermaNet data and computing semantic relatedness. In *Companion Volume of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'2005)*, Michigan, Ann Arbor, USA, June 2005.
- [Hamp and Feldweg 1997] B. Hamp and H. Feldweg. GermaNet - a Lexical-Semantic Net for German. In *Proceedings of ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, Madrid, Spain, 1997.

- [Hataoka et al. 2004] N. Hataoka, Y. Obuchi, T. Mitamura, E. Nyberg. Robust Speech Dialog Interface for Car Telematics Service. In *Proceedings of IEEE Consumer Communications and Networking Conference (CCNC2004)*, Las Vegas, USA, 2004.
- [Hovy 1988] E. H. Hovy. *Generating Natural Language under Pragmatic Constraints*. Lawrence Erlbaum Associates, Inc., 1988.
- [Hovy 1997] E. H. Hovy. Language Generation: Overview. In R. A. Cole, J. Mariani, H. Uszkoreit, G. Varile, A. Zaenen, V. Zue, and A. Zampolli (eds.). *Survey of the State of the Art in Human Language Technology, Studies in Natural Language Processing, Linguistica Computazionale*, Vol. XII-XIII, Cambridge University Press/Giardini Editori E Stampatori, Pisa, 1997.
- [Hunt 1997] M. J. Hunt. Signal Representation. In R. A. Cole, J. Mariani, H. Uszkoreit, G. Varile, A. Zaenen, V. Zue, and A. Zampolli (eds.). *Survey of the State of the Art in Human Language Technology, Studies in Natural Language Processing, Linguistica Computazionale*, Vol. XII-XIII, Cambridge University Press/Giardini Editori E Stampatori, Pisa, 1997.
- [Hutchens and Alder 1998] J. L. Hutchens and M. D. Alder. Introducing MegaHal. In D. M. W. Powers (ed). *NeMLaP3/CoNLL98 Workshop on Human Computer Conversation*, ACL, pp. 271-274, 1998.
- [Ide and Véronis 1998] N. Ide and J. Véronis. Introduction to the special issue on word sense disambiguation: the state of the art. *Computational Linguistics*, 24(1): 1-40, 1998.
- [Jameson 2001] A. Jameson. Modeling Both the Context and the User. *Personal and Ubiquitous Computing*, 5(1): 29-33, 2001.
- [Jameson 2003] A. Jameson. Adaptive interfaces and agents. In J. A. Jacko and A. Sears (eds). *Human-computer interaction Handbook*, pp. 305-330, Mahwah, NJ: Erlbaum, 2003.
- [Jameson et al. 2002] A. Jameson, J. Konstan, and J. Riedl. AI Techniques for Personalized Recommendation. *Tutorial at the Eighteenth National Conference on Artificial Intelligence (AAAI 02)*, Edmonton, Alberta, Canada, July 2002.
- [Jiang and Conrath 1997] J. J. Jiang and D. W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the International Conference on Research in Computational Linguistics*, pp. 19-33, Taiwan, 1997.
- [Jörg 2003] M. Jörg. Die semantische Auswertung von Produktanforderungen mit Hilfe von GermaNet. *GLDV – Workshop des Arbeitskreises Lexikographie "Anwendungen des deutschen Wortnetzes in Theorie und Praxis"*, Tübingen, 2003.
- [Johnston et al. 2001] M. Johnston, S. Bangalore, and G. Vasireddy. MATCH: Multimodal access to city help. In *Automatic Speech Recognition and Understanding Workshop*, Trento, Italy, 2001.

- [Jurafsky and Martin 2000] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. New Jersey: Prentice-Hall, 2000.
- [Kass 1989] R. Kass. Student modeling in intelligent tutoring systems – implications for user modeling. In A. Kobsa and W. Wahlster (eds). *User Models in Dialog Systems*. Berlin, Heidelberg: Springer, pp. 386-410, 1989.
- [Kass and Finin 1988] R. Kass and T. Finin. Modeling the User in Natural Language Systems. *Computational Linguistics* 14 (3): 5-22, 1988.
- [Kießling 2005] W. Kießling. Preference Queries with SV-Semantics. *International Conference on Management of Data (COMAD 2005)*, Goa, India, Jan. 2005.
- [Klein et al. 2003] D. Klein, J. Smarr, H. Nguyen, and C. Manning. Named Entity Recognition with Character-Level Models. In *Proceedings of CoNLL-2003*, pp. 180-183, Edmonton, Canada, 2003.
- [Kobsa, 1990] A. Kobsa. Modeling The User's Conceptual Knowledge in BGP-MS, a User Modeling Shell System. *Computational Intelligence* 6:193–208, 1990.
- [Kobsa 2001] A. Kobsa. Generic User Modeling Systems. *User Modeling and User-Adapted Interaction* 11(1-2): 49-63, 2001.
- [Kobsa et al. 2001] A. Kobsa, J. Koenemann, and W. Pohl. Personalized Hypermedia Presentation Techniques for Improving Online Customer Relationships. *The Knowledge Engineering Review* 16(2): 111-155, 2001.
- [Kobsa and Pohl 1995] A. Kobsa and W. Pohl. The User Modeling Shell System BGP-MS. *User Modeling and User-Adapted Interaction* 4(2): 59-106, 1995.
- [Kobsa and Wahlster 1989] A. Kobsa and W. Wahlster (eds). *User Models in Dialog Systems*. Heidelberg, Berlin: Springer, 1989.
- [Koch 2002] M. Koch. Global Identity Management to Boost Personalization. In P. Schubert and U. Leimstoll (eds). *Proc. Research Symposium on Emerging Electronic Markets*, pp. 137 – 147, Basel, Schweiz, 2002.
- [Koch et al. 2002] M. Koch, G. Groh, C. Hillebrand, and N. Fremuth. *Mobile Support for Lifestyle Communities*. Arbeitspapier Nr. 34 des Lehrstuhls für Allg. und Ind. Betriebswirtschaftslehre (AIB), Technische Universität München, Nov. 2002.
- [Koch and Möslein 2003] M. Koch, K. Möslein. User Representation in E-Commerce and Collaboration Applications. In *Proc. 16th Bled eCommerce Conference*, Bled, Slovenia, June 2003.
- [Kohlmaier et al. 2001] A. Kohlmaier, S. Schmitt, and R. Bergmann. A Similarity-Based Approach to Attribute Selection in User-Adaptive Sales Dialogs. In D. W. Aha and I. Watson (eds). *Lecture Notes In Computer Science*, Vol. 2080: 306-320, London: Springer, 2001.

- [Komatani et al. 2003] K. Komatani, F. Adachi, S. Ueno, T. Kawahara, and H. G. Okuno. Flexible Spoken Dialogue System based on User Models and Dynamic Generation of VoiceXML Scripts. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialogue*, Sapporo, Japan, July 2003.
- [Kouchnir 2004] B. Kouchnir. A Machine Learning Approach to German Pronoun Resolution. In L. van der Beek, D. Genzel, and D. Midgley (eds). *ACL 2004: Student Workshop. Association for Computational Linguistics*, pp. 55-60, Barcelona, Spain, 2004.
- [Krulwich 1997] B. Krulwich. Lifestyle Finder: Intelligent user profiling using large-scale demographic data. *AI Magazine* 18(2): 37-45, 1997.
- [Kruse et al. 2005] P. M. Kruse, A. Naujoks, M. Kunze, and D. Rösner. Clever Search: A WordNet Based Wrapper for Internet Search Engines. In B. Fisseni, H.-C. Schmitz, B. Schröder, and P. Wagner (eds). *Sprachtechnologie, mobile Kommunikation und linguistische Ressourcen: Applications of GermaNet II*, pp. 367-380, Frankfurt am Main: Peter Lang, 2005.
- [Kruskal 1999] J. B. Kruskal. An overview of sequence comparison. In D. Sankoff and J. Kruskal (eds). *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, Chapter 1, CSLI Publications, 1999.
- [Kuflik et al. 2005] T. Kuflik, C. Callaway, D. Goren-Bar, C. Rocchi, O. Stock, and M. Zancanaro. Non-Intrusive User Modeling for a Multimedia Museum Visitors Guide System. *10th International Conference on User Modeling (UM'05)*, Edinburgh, UK, 2005.
- [Kunze and Lemnitzer 2002a] C. Kunze and L. Lemnitzer. GermaNet - representation, visualization, application. In *Proc. LREC, main conference*, Vol. V: 1485-1491, 2002.
- [Kunze and Lemnitzer 2002b] C. Kunze and L. Lemnitzer. Standardizing Wordnets in a Web-compliant Format: The Case of GermaNet. In *Proc. LREC. Workshop on Wordnet Structures and Standardisation, and how these affect Wordnet Applications and Evaluations*, pp. 24-29, 2002.
- [Kunze and Lemnitzer 2005] C. Kunze and L. Lemnitzer. Die Zukunft der Wortnetze oder die Wortnetze der Zukunft - ein ROADMAP-Beitrag. In B. Fisseni, H.-C. Schmitz, B. Schröder, and P. Wagner (eds). *Sprachtechnologie, mobile Kommunikation und linguistische Ressourcen: Applications of GermaNet II*, pp. 339-352, Frankfurt am Main: Peter Lang, 2005.
- [Kunze and Rösner 2003] M. Kunze and D. Rösner. Issues in Exploiting GermaNet as a Resource in Real Applications. In *Proceedings of the GermaNet-Workshop: Anwendungen des deutschen Wortnetzes in Theorie und Praxis (organized by GLDV Arbeitskreis Lexikographie)*, Tübingen, Oct. 2003.
- [Langer and Naumann 2003] H. Langer and S. Naumann. Parsing natürlicher Sprache. In G. Görz, C.-R. Rollinger, and J. Schneeberger (eds). *Handbuch der Künstlichen Intelligenz*, Oldenbourg, 2003.

- [Langley 1997] P. Langley. Machine Learning for Adaptive User Interfaces. In *Proceedings of the 21st German Annual Conference on Artificial Intelligence*, pp. 53-62, Freiburg, Germany: Springer, 1997.
- [Larsen and Baeekgaard 1994] L. B. Larsen and A. Baeekgaard. Rapid prototyping of a dialogue system using a generic dialogue development platform. In *Proceedings of ICSLP'94*, pp. 919-922, Yokohama, Japan, 1994.
- [Larsson and Traum 2000] S. Larsson and D. Traum. Information state and dialogue management in the TRINDI Dialogue Move Engine Toolkit. *Natural Language Engineering* 6(3-4): 323-340, 2000.
- [Lavie et al. 2005] T. Lavie, J. Meyer, K. Bengler, and J. F. Coughlin. The Evaluation of In-Vehicle Adaptive Systems. S. Weibelzahl, A. Paramythis, and J. Masthoff (eds). In *Proc. of the Fourth Workshop on the Evaluation of Adaptive Systems*, held in conjunction with: 10th International Conference on User Modeling (UM'05), Edinburgh, UK , 2005.
- [Leckner et al. 2003] T. Leckner, M. Koch, R. Stegmann, and M. Lacher: Personalization meets Mass Customization – Support for the Configuration and Design of Individualized Products. In *Proc. of the International Conference on Enterprise Information Systems (ICEIS)*, Angers, France, 2003.
- [Leckner et al. 2004] T. Leckner, R. Stegmann, and J. Schlichter. Reducing Complexity for Customers by means of a Model-Based Configurator and Personalized Recommendations, In *Proc. of the International Conference on Economic, Technical and Organisational aspects of Product Configuration Systems*, Lyngby, Denmark, 2004.
- [Lekakos and Giaglis 2002] G. Lekakos and G. Giaglis. Delivering personalized advertisements in digital television: A methodology and empirical evaluation. In *Proceedings of the Personalization in Future TV Workshop*, pp. 119-129, 2002.
- [Lesk 1986] M. Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proc. of the 5th Annual international Conference on Systems Documentation (SIGDOC '86)*, Toronto, Ontario, Canada, 1986.
- [Linden et al. 1997] G. Linden, S. Hanks, and N. Lesh. Interactive assessment of user preference models: The automated travel assistant. In A. Jameson, C. Paris, and C. Tasso (eds). *User modeling: Proceedings of the Sixth International Conference, UM97*, pp. 67-78, Vienna: Springer, 1997.
- [Litman and Pan 2002] D. J. Litman and S. Pan. Designing and Evaluating an Adaptive Spoken Dialogue System. *User Modeling and User-Adapted Interaction* 12: 111-137, Kluwer Academic Publishers, 2002.
- [Magnini and Strapparava 2001] B. Magnini and C. Strapparava. Using WordNet to Improve User Modelling in a Web Document Recommender System. In *Proceedings of the NAACL Workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations*, pp. 132-137, Pittsburgh, PA, 2001.
- [Manning and Schütze 1999] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press, 1999.

- [McDonald 1983] D. D. McDonald. Natural Language Generation as a Computational Problem: An Introduction. In M. Brady and R. Berwick (eds). *Computational Models of Discourse*. Cambridge, MA: MIT Press, 1983.
- [McGlashan et al. 1990] S. McGlashan, E. Bilange, N. Fraser, N. Gilbert, P. Heisterkamp, and N. Youd. *Managing oral dialogues*. Research Report, Social and Computer Sciences Research Group, University of Surrey, Surrey, UK, 1990.
- [McKeown 1985] K. R. McKeown. *Text generation: using discourse strategies and focus constraints to generate natural language text*. Cambridge University Press, 1985.
- [McNee et al. 2003] S. M. McNee, S. K. Lam, J. Konstan, and J. Riedl. Interfaces for Eliciting New User Preferences in Recommender Systems. In *Proceedings of The 9th International Conference on User Modeling (UM'2003)*, pp. 178-188, June 2003.
- [McRoy 1992] S. W. McRoy. Using multiple knowledge sources for word sense discrimination. *Computational Linguistics* 18(1): 1-30, 1992.
- [McTear 2002] M. F. McTear. Spoken dialogue technology: enabling the conversational user interface. *ACM Comput. Surv.* 34(1): 90-169, March 2002.
- [Mehta et al. 2005] B. Mehta, C. Niederée, A. Stewart, M. Degemmis, P. Lops, and G. Semeraro. Ontologically-Enriched Unified User Modeling for Cross-System Personalization. In *Proc. of the 10th International Conference on User Modeling*, Edinburgh, UK, 2005.
- [Melville et al. 2002] P. Melville, R. Mooney, and R. Nagarajan. Content-Boosted Collaborative Filtering for improved recommendations. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02)*, July 2002.
- [Mertens and Höhl 1999] P. Mertens and M. Höhl. Wie lernt der Computer den Menschen kennen? – Experimente zur Benutzermodellierung in der Wirtschaftsinformatik. *Wirtschaftsinformatik* 41(3), 1999.
- [Miettinen et al 2002] M. Miettinen, P. Nokelainen, J. Kurhila, T. Silander, and H. Tirri. Adaptive Profiling Tool for Teacher Education. In *Proceedings of the SITE 2002 Conference*, pp. 1153-1157, Nashville, USA, March 2002.
- [Miller et al. 2003] B. Miller, I. Albert, S. Lam, J. Konstan, and J. Riedl. MovieLens Unplugged: Experiences with a Recommender Systems on Four Mobile Devices. In *Proc. of the 2003 ACM Conference on Intelligent User Interfaces*, Miami, FL, Jan. 2003.
- [Miller 1995] G. A. Miller. WordNet: A lexical database for English. *Communications of the ACM* 38 (11): 39 – 41, November 1995.
- [Minker et al. 2003] W. Minker, U. Haiber, P. Heisterkamp, and S. Scheible. Intelligent Dialog Overcomes Speech Technology Limitations: The SENECa Example. In *Proc. International Conference on Intelligent User Interfaces (IUI)*, Miami, FL, 2003.

- [Mitchell et al. 1994] P. Mitchell, M. Santorini, B. Santorini, and M. A. Marcinkiewicz. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics* 19(2): 313-330, 1994.
- [Montoyo et al. 2005] A. Montoyo, A. Suarez, G. Rigau, and M. Palomar. Combining Knowledge- and Corpus-based Word-Sense-Disambiguation Methods. *Journal of Artificial Intelligence Research* Vol. 23: 299-330, 2005.
- [Navigli and Velardi 2005] R. Navigli and P. Velardi. Structural Semantic Interconnections: A Knowledge-Based Approach to Word Sense Disambiguation. *IEEE Trans. Pattern Anal. Mach. Intell.* 27(7): 1075-1086, July 2005.
- [O'Connor et al. 2001] M. O'Connor, D. Cosley, J. Konstan, and J. Riedl. PolyLens: A Recommender System for Groups of Users. In *Proceedings of the European Conference on Computer-Supported Cooperative Work (ECSCW'01)*, Bonn, Germany, 2001.
- [O'Nuallain and Redfern 2005] C. O'Nuallain and S. Redfern. Providing more effective curriculum through building dynamic profiles and tracking user behaviour. In *iNEER Conf. for Engineering, Education, and Research (iCEER-2005)*, Tainan, Taiwan, March 2005.
- [Pandzic 2001] I. S. Pandzic. Life on the Web. *Software Focus* 2(2): 52-59, 2001.
- [Pazzani et al. 1996] M. Pazzani, J. Muramatsu, and D. Billsus. Syskill & Webert: Identifying interesting web sites. In *Proceedings of the National Conference on Artificial Intelligence*, Portland, OR, 1996.
- [Piller 2001] F. Piller. *Mass Customization. Ein wettbewerbsstrategisches Konzept im Informationszeitalter*. Wiesbaden: Gabler Deutscher Universitäts-Verlag, 2001.
- [Pine 1993] J. Pine. *Mass Customization: The New Frontier in Business Competition*. Boston: Harvard Business School Press, 1993.
- [Pohl et al. 1995] W. Pohl, A. Kobsa, and O. Kutter. User Model Acquisition Heuristics Based on Dialogue Acts. In *International Workshop on the Design of Cooperative Systems*, pp. 471-486, Antibes-Juan-les-Pins, France, 1995.
- [Pollard and Sag 1994] C. J. Pollard and I. A. Sag. *Head-Driven Phrase Structure Grammar*. Chicago, University of Chicago Press, 1994.
- [Pretschner and Gauch 1999] A. Pretschner and S. Gauch. Ontology Based Personalized Search. In *Proc. 11th IEEE Intl. Conf. on Tools with Artificial Intelligence*, pp. 391-398, Chicago, November 1999.
- [Rashid et al. 2002] A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. McNee, J. Konstan, and J. Riedl. Getting to know you: Learning new user preferences in recommender systems. In *Proc. 7th Int. Conf. on Intelligent User Interfaces (IUI'02)*, San Francisco, California, 2002.
- [Raskutti and Zukerman 1997] B. Raskutti and I. Zukerman. Generating Queries and Replies during Information-seeking Interactions. *International Journal of Human Computer Studies* 47(6): 689-734, 1997.

- [Razmerita et al. 2003] L. Razmerita, A. Angehrn, and A. Maedche. Ontology-based User Modeling for Knowledge Management Systems. In *Proc. of the 9th International Conference on User Modeling*, University of Pittsburgh, Johnstown, USA, 2003.
- [Reichwald et al. 2001] R. Reichwald, N. Fremuth, and M. Ney. COSMOS – Teilprojekt Lifestyle Community. In U. Baumgarten, H. Krcmar, R. Reichwald, and J. Schlichter (eds). *Community Online Services and Mobile Solutions – Projektstartbericht des Verbundvorhabens COSMOS*, TUM-Bericht I-0105, pp. 4-16, München 2001.
- [Reiter et al. 1997] E. Reiter, A. Cawsey, L. Osman, and Y. Roff. Knowledge Acquisition for Content Selection. In *Proceedings of the 6th European Workshop on Natural Language Generation*, pp. 117-126, Duisburg, Germany, 1997.
- [Reiter and Dale 2000] E. Reiter and R. Dale. *Building Natural-Language Generation Systems*. Cambridge University Press, 2000.
- [Reitter et al. 2001] D. Reitter, S. Covaci, F. Oltean, C. Bacanu, and T. Serbanuta. Hybrid natural language processing in a customer-care environment. In *Proc. of the 11th TaCoS*, 2001.
- [Renneberg and Borghoff 2003] V. Renneberg and U. M. Borghoff. Pipelined Filter Combination in Product Personalization. In C. Stephanidis (ed). In *Proc. 10th International Conf. on Human-Computer Interaction (HCI 2003)*, Crete, Greece, 2003. Mahwah, NJ: Lawrence Erlbaum Ass. Publ., Vol. 4, 2003.
- [Renneberg et al. 2004] V. Renneberg, T. Leckner, and R. Stegmann. Recommending Personalized Products Based on Product Scorings. In *Proc. of the Workshop on Configuration 2004 held in Conjunction with 16th European Conference on Artificial Intelligence (ECAI-04)*, Valencia, Spain, 2004.
- [Resnick et al. 1994] P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 Computer Supported Collaborative Work Conference*, 1994.
- [Resnik and Diab 2000] P. Resnik and M. Diab. Measuring Verb Similarity. In *Proc. Twenty Second Annual Meeting of the Cognitive Science Society (COGSCI2000)*, Philadelphia, USA, Aug. 2000.
- [Rich 1979] E. Rich. User modeling via stereotypes. *International Journal of Cognitive Science* Vol. 3:329-354, 1979.
- [Rist et al. 2003] T. Rist, E. André, S. Baldes, P. Gebhard, M. Klesen, M. Kipp, P. Rist, and M. Schmitt. A Review of the Development of Embodied Presentation Agents and Their Application Fields. In H. Prendinger and M. Ishizuka (eds). *Life-Like Characters: Tools, Affective Functions, and Applications*, pp. 377-404, Springer, 2003.
- [Rogers 1999] S. Rogers. An Adaptive Interactive Agent for Route Advice. In *Proc. of the Third Annual Conf. on Autonomous Agents*, pp. 198-205, Seattle, Washington, 1999.

- [Sabin and Weigel 1998] D. Sabin and R. Weigel. Product Configuration Frameworks – A Survey. *IEEE Intelligent Systems and their applications* 13(4): 42-49, 1998.
- [Sadek et al. 1997] M. D. Sadek, P. Bretier, and F. Panaget. ARTIMIS: Natural dialogue meets rational agency. In *Proc. of 15th Int. Joint Conference on Artificial Intelligence (IJCAI-97)*, San Francisco, CA, Morgan Kaufmann Publishers, pp. 1030-1035, 1997.
- [Sarwar et al. 2000] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of Dimensionality Reduction in Recommender System – A Case Study. In *ACM WebKDD 2000 Web Mining for E-Commerce Workshop*, 2000.
- [Schafer et al. 2000] J. B. Schafer, J. Konstan, and J. Riedl. Electronic Commerce Recommender Applications. *Journal of Data Mining and Knowledge Discovery* 5(1/2): 115-152, 2000.
- [Schein et al. 2002] A. Schein, A. Popescul, L. Ungar, and D. Pennock. Methods and Metrics for Cold-Start Recommendations. In *Proceedings of the 25th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002)*, 2002.
- [Schiller et al. 1995] A. Schiller, S. Teufel, and C. Thielen. *Guidelines für das Tagging deutscher Textcorpora mit STTS*. Technical report, IMS Stuttgart/Seminar f. Sprachwiss., Tübingen, 1995.
- [Schmid 1994] H. Schmid. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *Proc. of the International Conference on New Methods in Language Processing*, pp. 44-49, Manchester, UK, 1994.
- [Schmitt and Bergmann 2001] S. Schmitt and R. Bergmann. A Formal Approach to Dialogs with Online Customers. In B. O'Keefe, C. Loebbecke, J. Gricar, A. Pucihar, and G. Lenart (eds). *e-Everything: e-Commerce, e-Government, e-Household, e-Democracy. Proc. of the 14th Bled Electronic Commerce Conference*, pp. 309-328, Bled, Slovenia, 2001.
- [Schulze 1992] G. Schulze. *Die Erlebnisgesellschaft – Kulturosoziologie der Gegenwart*. Frankfurt: Campus Verlag, 1992.
- [Schwab and Kobsa 2002] I. Schwab and A. Kobsa. Adaptivity through Unobstrusive Learning. In *KI Vol. 3, Special Issue on Adaptivity and User Modeling*, 2002.
- [Seco et al. 2004] N. Seco, T. Veale, and J. Hayes. An Intrinsic Information Content Metric for Semantic Similarity in WordNet. In *Proceedings of ECAI'2004, the 16th European Conference on Artificial Intelligence*, Valencia, Spain, 2004.
- [Semeraro et al. 2005] G. Semeraro, M. Degemmis, P. Lops, and I. Palmisano. WordNet-based User Profiles for Semantic Personalization. In *PIA 2005 – Workshop on New Technologies for Personalized Information Access*, pp. 74-83, Edinburgh, UK, 2005.
- [Smith and Hipp 1994] R. W. Smith and D. R. Hipp. *Spoken Natural Language Dialog Systems*. New York, Oxford: Oxford University Press, 1994.

- [Spellerberg and Berger-Schmitt 1998] A. Spellerberg and R. Berger-Schmitt. *Lebenstile im Zeitvergleich. Typologien für West- und Ostdeutschland 1993 und 1996*. Discussion Paper FS-III 98-403, Berlin: Wissenschaftszentrum, Berlin 1998.
- [Staab et al. 1999] S. Staab, C. Braun, I. Bruder, A. Düsterhöft, A. Heuer, M. Klettke, G. Neumann, B. Prager, J. Pretzel, H.-P. Schnurr, R. Studer, H. Uszkoreit, and B. Wrenger. GETESS - Searching the Web Exploiting German Texts. In M. Klusch, O. M. Shehory, and G. Weiß (eds). *Lecture Notes in Artificial Intelligence, 3rd International Workshop on Cooperative Information Agents (CIA '99)* Vol. 1652: 113-124, Springer, Uppsala, Sweden, 1999.
- [Stegmann 2005] R. Stegmann. Improving Explicit Profile Acquisition by means of Adaptive Natural Language Dialog. In *Proc. of the 10th International Conference on User Modeling*, Edinburgh, UK. Berlin, Heidelberg: Springer, LNAI 3538, pp. 518-520, 2005.
- [Stegmann et al. 2003] R. Stegmann, M. Koch, M. Lacher, T. Leckner, and V. Renneberg. Generating Personalized Recommendations in a Model-Based Product Configurator System. *Workshop on Configuration, International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, Aug. 2003.
- [Stegmann et al. 2004] R. Stegmann, M. Koch, and W. Wörndl. Acquisition of Customer Profiles by means of Adaptive Text-Based Natural Language Dialog. In *Proceedings of the Annual Workshop of the SIG Adaptivity and User Modeling in Interactive Systems of the German Informatics Society (GI)(ABIS04)*, Berlin, Germany, 2004.
- [Stegmann and Wörndl 2005] R. Stegmann and W. Wörndl. Using GermaNet to Generate Individual Customer Profiles. In B. Fisseni, H.-C. Schmitz, B. Schröder, and P. Wagner (eds). *Sprachtechnologie, mobile Kommunikation und linguistische Ressourcen: Applications of GermaNet II*, pp. 381-393, Frankfurt am Main: Peter Lang, 2005.
- [Stolze and Ströbel 2001] M. Stolze and M. Ströbel. Utility-Based Decision Tree Optimization: A Framework for Adaptive Interviewing. In *Proc. of the 8th international Conference on User Modeling 2001*. M. Bauer, P. J. Gmytrasiewicz, and J. Vassileva (eds). Lecture Notes In Computer Science, Vol. 2109: 105-116, London: Springer, 2001.
- [Stolze and Ströbel 2004] M. Stolze and M. Ströbel. Recommending as Personalized Teaching: Towards Credible Needs-based eCommerce Recommender Systems. In C.-M. Karat, J. O. Blom, J. Vanderdonekt, and J. Karat (eds). *Designing Personalized User Experience*, pp. 293-314, Dordrecht: Kluwer Academic Publishers, 2004.
- [Thompson et al. 2004] C. A. Thompson, M. Göker, and P. Langley. A personalized system for conversational recommendations. *Journal of Artificial Intelligence Research* Vol. 21: 393-428, 2004.
- [Towle and Quinn 2000] B. Towle and C. Quinn. Knowledge Based Recommender Systems Using Explicit User Models. *The AAAI-2000 Workshop on Knowledge-based Electronic Markets (KBEM'00)*, Austin, TX, USA , 2000.
- [Tseng and Piller 2003] M. Tseng and F. Piller. The Customer Centric Enterprise. In M. Tseng and F. Piller (eds), *The Customer Centric Enterprise: Advances in Mass Customization and Personalization*, pp. 1-18, New York, Berlin: Springer, 2003.

- [Wärnestål 2005] P. Wärnestål. Modularized User Modeling in Conversational Recommender Systems. In *Proc. of the 10th International Conference on User Modeling*, Edinburgh, UK. Berlin, Heidelberg: Springer, LNAI 3538, pp. 545–547, 2005.
- [Wahlster 2000] W. Wahlster (ed). *Verbmobil: Foundations of Speech-to-Speech Translation*. Berlin, Heidelberg, New York, Barcelona, Hong Kong, London, Milan, Paris, Singapore, Tokyo: Springer, July 2000.
- [Wahlster et al. 2001] W. Wahlster, N. Reithinger, and A. Blocher. SmartKom: Multimodal Communication with a Life-Like Character. In *Proc. of Eurospeech 2001, 7th European Conf. on Speech Communication and Technology*, Vol. 3: 1547-1550, Aalborg, Denmark, 2001.
- [Ward 1989] W. Ward. Modelling Non-Verbal Sounds for Speech Recognition, In *Proc. DARPA Workshop on Speech and Natural Language*, pp. 47-50, 1989.
- [Ward and Pellom 1999] W. Ward and B. Pellom. The CU Communicator system. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*, Keystone, CO, 1999.
- [Weiser 1993] M. Weiser. Hot Topics: Ubiquitous Computing. *IEEE Computer*, Oct. 1993.
- [Weizenbaum 1966] J. Weizenbaum. ELIZA - A Computer Program for the Study of Natural Language Communication between Man and Machine. *Communications of the Association for Computing Machinery* 9: 36-45, 1966.
- [Wörndl 2003] W. Wörndl. *Privatheit bei dezentraler Verwaltung von Benutzerprofilen*. Doctoral Thesis, Technische Universität München, Aug. 2003.
- [Zadrozny et al. 2000] W. Zadrozny, M. Budzikowska, J. Chai, N. Kambhatla, S. Levesque, and N. Nicolov. Natural Language Dialog for Personalized Interaction. *Communications of ACM* 43(8): 116-120, 2000.
- [Zue and Cole 1997] V. Zue and R. Cole. Spoken Language Input. In R. A. Cole, J. Mariani, H. Uszkoreit, G. Varile, A. Zaenen, V. Zue, and A. Zampolli (eds.). *Survey of the State of the Art in Human Language Technology, Studies in Natural Language Processing, Linguistica Computazionale*, Vol. XII-XIII, Cambridge University Press/Giardini Editori E Stampatori, Pisa, 1997.
- [Zue and Glass 2000] V. Zue and J. Glass. Conversational Interfaces: Advances and Challenges. In *Proc. of the IEEE* 88(8): 1166-1180, Aug. 2000.
- [Zukerman and Litman 2001] I. Zukerman and D. Litman. Natural Language Processing and User Modeling: Synergies and Limitations. *User Modeling and User-Adapted Interaction* Vol. 11: 129-158, 2001.