

Carnegie Mellon



Universität Karlsruhe (TH)
Research University • founded 1825

Rapid Simulation-Driven Reinforcement Learning of Multimodal Dialog Strategies for Human-Robot Interaction

Diploma Thesis

May 2006



Carnegie Mellon University, USA
Universität Karlsruhe (TH), Germany

Interactive Systems Laboratories

cand. inform.

Thomas Prommer

Supervisors:

Prof. Dr. Alexander Waibel
Dipl.-Inform. Hartwig Holzapfel
Dr.-Ing. Thomas Schaaf

Declaration

English:

Herewith I confirm that I independently prepared this diploma thesis. No further auxiliary means as the ones indicated in this work were used for the preparation.

Deutsch:

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

May 31st, 2006



Thomas Prommer

Abstract

In this work, we propose a procedure model for rapid automatic strategy learning in multimodal human-robot dialogs. Our approach is tailored for typical task-oriented and slot-based human-robot dialogs, with no prior knowledge about the expected user dynamics and system performance metrics being present. Given such scenario, we propose the use of data-driven dialog simulation, where the necessary user and system error models are solely trained through the initial execution of an inexpensive Wizard-of-Oz experiment. In order to challenge data sparsity, we fall back on the use of low-conditioned probabilistic methods for user and error modeling. We claim that for the addressed types of dialogs, often already a small Wizard-of-Oz dialog corpus suffices to train the simulation model in such accurate manner, that it provides artificial dialog experience from which powerful dialog strategies can be learned through reinforcement learning. If desired, such rapidly learned initial strategies can be continuously improved or respectively adapted to environmental changes by retraining the simulation model based on later available online experience. We empirically validate our approach by applying our procedure model to an authentic exemplary human-robot dialog scenario. Within our strategy evaluation, we do not only rely on the convincing performance of the learned strategy in the simulation environment, but also show its supremacy to a non-trivial handcrafted baseline strategy in the real world experiment. As a key novel contribution, to the author's knowledge, this work is the first to perform automatic learning not only of speech-based, but of multimodal dialog strategies.

Acknowledgement

This diploma thesis was performed within the interACT exchange program in equal extent at Carnegie Mellon University, Pittsburgh, USA and University of Karlsruhe, Germany. In this context, I want to thank the advisory board of the interACT program for granting me their scholarship and thereby the opportunity to perform my research in two scientifically exciting locations. I am also grateful to the Baden-Württemberg scholarship association for sponsoring my stay abroad.

I further would like to extend my thanks to my first advisor, Hartwig Holzapfel, who substantially supported this research with his invaluable scientific contributions. His pleasant manner of sedulously supervising this work's course without restricting my chosen direction was a pleasant experience. I also want to thank Andy Barto for his inspiring teachings of reinforcement learning at University of Massachusetts, Amherst. His contagious enthusiasm for this young discipline attracted my immediate interest and ultimately resulted in my decision to pursue this thesis.

Further thanks are extended to Kai Nickel and Pedram Azad for their unwavering support within the setup phase of the experimental human-robot system. The numerous test subjects, who volunteered at Carnegie Mellon University and University of Karlsruhe, are also recognized. For without their contribution, this work would not have been realizable.

Finally, I would like to offer my gratitude to the four people closest to me for their constant encouragement and shown understanding during the conduction of this thesis.

Table of Contents

1	INTRODUCTION	1
1.1	Motivation	1
1.2	Contributions	3
1.3	Contents of this work.....	4
2	BASICS	7
2.1	Human-machine dialogs	7
2.1.1	Dialog application types	8
2.1.2	General architecture and layers of a spoken dialog system	10
2.2	Markov decision processes and reinforcement learning	13
2.2.1	Markov decision processes.....	13
2.2.2	Reinforcement learning.....	15
2.3	Reinforcement learning of dialog strategies	26
2.3.1	Formulating dialog as a Markov decision process.....	27
2.3.2	Model-based vs. simulation-based strategy learning	28
2.3.3	Dialog Simulation	31
3	RELATED WORK	35
3.1	Reinforcement learning of dialog strategies.....	35
3.1.1	Model-based strategy learning	35
3.1.2	Simulation-based strategy learning.....	37
3.2	Dialog simulation techniques	39
3.2.1	User modeling	40

3.2.2	ASR error modeling.....	44
4	PROCEDURE MODEL.....	47
4.1	Introduction.....	47
4.2	Step 1: Conduct a Wizard-of-Oz experiment.....	48
4.3	Step 2: Train simulation model.....	52
4.3.1	Simulation architecture	53
4.3.2	User model.....	54
4.3.3	ASR error model	56
4.3.4	Error modeling of modalities other than speech.....	58
4.4	Step 3: Definition of the MDP	58
4.5	Step 4: Learning the strategy.....	60
4.6	Step 5: Strategy deployment	61
4.7	Step 6: Retrain simulation	62
5	EXPERIMENT	65
5.1	The task.....	65
5.1.1	Introduction and background	65
5.1.2	Task description	66
5.1.3	The dialog setup.....	68
5.2	Wizard-of-Oz experiment	70
5.2.1	Experimental setup.....	70
5.2.2	Experiment execution.....	72
5.2.3	User interviews	73
5.2.4	Additional data collection	75
5.3	Training of the simulation.....	75
5.3.1	User model.....	76
5.3.2	ASR error model	76
5.3.3	3D pointing gesture recognizer error model	77
5.4	The MDP design	79
5.4.1	Reward function.....	79
5.4.2	Action-state space design	79
5.5	Strategy learning.....	82
5.6	Experimental validation	82
5.7	Simulation retraining	83

6	RESULTS AND DISCUSSION	87
6.1	Experiments within the simulation domain	87
6.1.1	Evaluating different state-space configurations	87
6.1.2	Finding the best λ value	90
6.1.3	State features and their effect on the optimal strategy	91
6.2	Experimental validation results.....	93
6.3	Simulation retraining results	96
7	CONCLUSION AND FURTHER WORK.....	99
7.1	Summary and conclusion.....	99
7.2	Further work	101
A	HANDOUTS	103
B	SAMPLE DIALOGS	106
B.1	Wizard-of-Oz dialog example	106
B.2	Dialog examples of online-operation using the baseline strategy	107
B.3	Dialog examples of online-operation using the learned strategy	109
C	SIMULATION RETRAINING	112
	BIBLIOGRAPHY	113

Figure index

Figure 2.1 General architecture and layers of a spoken dialog system (SDS).....	10
Figure 2.2 Sequential decision processes.....	14
Figure 2.3 Model-based approach for strategy learning	29
Figure 2.4 Simulation-based approach for strategy learning	30
Figure 2.5 Typical dialog simulation work flow	32
Figure 4.1 Developed procedure model in a step by step manner.....	48
Figure 4.2 General architecture of the developed dialog simulation framework	53
Figure 5.1 SFB588 robot ARMAR.....	66
Figure 5.2 Photograph of the experimental setup.....	66
Figure 5.3 Visualization of a video frame within the 3D world of the gesture recognizer component [40].....	68
Figure 5.4 Dialog system automaton	69
Figure 5.5 Photograph of the Wizard-of-Oz experiment (human robot (left), test user (right))	71
Figure 5.6 Sketch of the experimental setup of the Wizard-of-Oz experiment	72
Figure 5.7 Evaluation graphs of the performed user interviews	74
Figure 5.8 Normal distribution of observed error angles of the recognized gestures	78
Figure 6.1 Strategy evaluation of different state-space configurations.....	88
Figure 6.2 Evaluation of different Lambda values.....	90
Figure 6.3 Action frequency within the optimal strategy for the speech condition feature.....	91
Figure 6.4 Action frequency within the optimal strategy for the ASR history feature	92
Figure 6.5 Action frequency within the optimal strategy for the gesture recognition condition feature.....	93
Figure 6.6 Action frequency of the optimal strategy before and after simulation retraining	97

Table index

Table 5.1 Available system actions and their preconditions	70
Table 5.2 Bi-gram user model learned from Wizard-of-Oz dialog corpus.....	76
Table 5.3 ASR error model for close speech trained on the Wizard-of-Oz dialog corpus.....	77
Table 5.4 ASR error model for distant speech trained on the Wizard-of-Oz dialog corpus.....	77
Table 5.5 3D error model for pointing gesture recognizer	78
Table 5.6 User model trained on the online-operation dialog corpus.....	84
Table 5.7 Close speech ASR error model trained on the online-operation dialog corpus.....	84
Table 6.1 Comparison of strategy performance using different state-space configurations	88
Table 6.2 Comparative study of the baseline strategy and the learned strategy	94
Table C.1 User model trained on merged dialog corpus of WOz experiment and online-operation corpus.....	112
Table C.2 Close speech ASR error model trained on merged dialog corpus of WOz experiment and online-operation corpus.....	112
Table C.3 3D pointing gesture recognizer error model used for simulation retraining	112

Abbreviations

ASR	Automatic speech recognition
DM	Dialog manager
HMM	Hidden Markov model
MDP	Markov decision process
MSDS	Multimodal spoken dialog systems
NLG	Natural language generation
NLU	Natural language understanding
RL	Reinforcement learning
RR	Recognition rate
RT	Recognition task
SC	Semantic concept
SDP	Sequential decision process
SDS	Spoken dialog system
TD	Temporal-difference
TFS	Typed feature structure
TTS	Text-to-Speech
WOz	Wizard-of-Oz

1 Introduction

1.1 Motivation

The endeavor of building autonomous systems which facilitate communication with machines in a natural way is well observed throughout history. Already in the 19th century, Alexander Graham Bell strived for the automatic processing of the most intuitive and efficient communication modality of humanity: the human speech. His so called phonoautograph was intended to support deaf people's lives by transcribing spoken words into text. While he never saw his creation brought to life, a few years later Bell's research was channeled into creating an invention that to this day is the most popular means of distant communication: the telephone.

Since then, researchers have continuously focused their efforts on designing intuitive human-machine dialog systems to facilitate ease and use of arbitrary electronic systems, independent of the user's level of expertise, age, or physical capabilities. Thanks to significant improvements in the domain of automatic speech recognition, natural language processing, and text-to-speech synthesis, sophisticated human-machine dialog systems have become a reality and are nowadays common in our every day life. In fact, human-machine dialog systems these days experience a so far unsighted level of attention within the research domain as well as widespread industry sectors¹. Particularly telephone-based spoken dialog systems are widely deployed within all imaginable areas of the commercial landscape, replacing human-driven call centers especially due to the immense economical savings their application promises. Consequently, whether you attempt to rebook your business flight last minute, or you want to get up-to-date information about the evening plays

¹ Google Inc. © announced last month (04/2006) working under high pressure on a speech-enabled interface for their market-leading search engine, targeted for the usage through mobile devices.

listed at your local cinema, your already rare chance of being greeted by a human voice decreases day to day.

Such mostly telephone-based information retrieval systems, however, highlight only one application of human-machine dialog systems. Another application environment, which will be focused within this thesis, is the domain of humanoid robots. Unlike rigid and rather impersonal robotic systems used in the industrial environment, humanoid robots are designed manlike with respect to their physical appearance and tailored to be embedded within the human's day-to-day life. The main target of application for these robots is the commanded or proactive support, or even takeover, of bothersome, everyday, human tasks such as household chores. Hereby, frequent communication between the robot and its human operators is required. Considering the intended human-like design of such robot systems, also within the field of communication an approximation of typical human dialog capabilities is desired. Consequently, the need of the development and deployment of highly natural, intelligent, and intuitive human-robot dialog systems arises. Hereby, the desired approximation of typical human face-to-face interaction however involves substantially more than the simple processing of speech. Instead, within the domain of humanoid robots, system designers recognize the need for building multimodal dialog systems capable of perceiving and understanding all semantic-carrying modalities also used by humans, such as for example pointing gestures, face expressions or body language.

Independent of their application domain or addressed modalities, dialog systems share a common characteristic in that they are usually complex and time-consuming to develop. Often, reusability potential of earlier deployed systems is very restricted as application environment and dialog context of a dialog system are usually unique. Correspondingly, the receptivity of direct porting processes of developed components from other dialog systems is inhibited. Especially the necessary design tasks within the domain of dialog management represent a crucial and often tedious undertaking within the development process. In order to design an intelligently acting dialog system, the so called dialog strategy specifying the system's behavior has to be thoroughly tailored for the particular task and existent dialog dynamics. Hereby, an often exhaustive mapping of possible dialog states to adequate system actions has to be specified. The appropriateness of such designed decision-making process is an essential quality factor for the dialog system as it significantly influences the success, efficiency, and naturalness of the dialog interaction. However, considering the uncertainty of specific user dynamics as well as the existence of error-prone system components, the process of designing an adequate dialog strategy is multifaceted and non-trivial. And yet, despite its importance and complexity, dialog strategies typically are hand-crafted and based on the intuition of a human expert designer. A problem that hereby arises is that, depending on the complexity of the dialog system, handcrafted strategy design easily becomes a time-consuming, sometimes even intractable affair. Also, key design criteria for a well performing strategy are often subtle and hidden within the dialog systems and can ultimately escape the attention of a human designer.

As an alternative to a human and handcrafted strategy design, the machine learning technique reinforcement learning (RL) has become popular for automatic dialog strategy learning. Essentially, hereby the dialog interaction is mapped to a finite action-state representation, a so called Markov decision process (MDP). Given a fixed number of state features, the current dialog state is described, while the execution of system actions, including the corresponding reaction of the environment (e.g. user answer and system processing), leads to state transitions. For such formalism, assuming the presence of sufficient knowledge or experience about the environment, the optimal system strategy with respect to a designed reward function can be automatically learned.

This thesis will have its focus on such introduced methodology of automatic strategy learning. Diverging from earlier work in this domain, in which strategy learning was performed for already deployed telephone-based systems, this thesis will introduce a procedure model for automatic strategy learning during system development within the domain of multimodal human-robot dialog interactions. The more precise contributions and environment of this work will be introduced in the subsequent subchapter.

1.2 Contributions

The overall contribution of this work is the presentation of a holistic procedure model to automatic strategy learning, tailored for the application within task-oriented human-robot dialog systems. The novel contributions of our work, in comparison to earlier work, hereby are multifaceted and located within different stages and levels of the automatic strategy learning process.

To date, automatic strategy learning has been applied to dialog systems at a point of time when the addressed systems were already operating within the real world for a significant period, using handcrafted strategies. In such work, the available online-operation experience was used as basis for applying reinforcement learning techniques, in order to replace the initial handcrafted strategies by superior learned strategies. Our approach differs from such work in a way that it intends strategy learning during the development of the dialog system. Thereby our approach contributes the possibility of using learned and optimized strategies from the first moment of online-operation and excludes completely the need of handcrafting strategies. Our approach neither requires prior online-operation nor any other experience about the dialog systems dynamics, but instead bases strategy learning on a rather small dialog corpus, collected through the execution of an initial Wizard-of-Oz study. We consider such a rapid approach to strategy learning especially within the domain of humanoid robots as sensible: Typical human-robot interactions involve the directives of a particular task from the user to the robot. Such mandates usually comprise ambiguities which require prompts for additional information, this time from the robot to the user. Our belief is that, given such straight-forward task-oriented and slot-based dialog design, often a small dialog corpus is already sufficient to learn powerful, at best optimal dialog strategies in such environment.

Hereby a key concept of our work is that we propose a simulation-driven approach to automatic strategy learning. For such dialog simulation setup, we adapt and extend earlier proposed probabilistic simulation techniques for user modeling and error modeling in a way that they fit the environmental conditions of our approach and the characteristic of human-robot interactions. Through the use of simplistic and low-conditioned probabilistic methods, we adequately address the restricted availability of training data on the one side, while also facilitating an easy and fast training process of the simulation model on the other side. In general, the use of our simplistic probabilistic models, combined with the proposed ease of data collection using a Wizard-of-Oz trial, intentionally promotes a comparatively rapid process of automatic strategy learning.

An important novel contribution of this work is the incorporation of the aspect of multimodality to strategy learning. To the author's knowledge, this work is the foremost research in which performance of automatic learning of multimodal dialog strategies has been examined. As introduced earlier, especially within the domain of humanoid robots, multimodal dialog interactions take a crucial and important role. Clearly, given a multimodal context, dialog complexity increases significantly compared to only speech-based systems. Correspondingly, the feasibility of handcrafted strategy design in such scenarios is even more questionable and automatic strategy learning becomes more beneficial. Recalling our simulation-driven approach to strategy learning, the incorporation of multimodality within strategy learning consequently also involves the necessity of multimodal dialog simulation. Consequently our work contributes a proposal for the design of a multimodal dialog simulation model, capable of simulating multimodal user actions as well as modality-specific error-proneness on the system side.

Ultimately, this work presents an empirical evaluation of our procedure model for a concrete and authentic human-robot interaction utilized within the SFB588 project of University of Karlsruhe²: After we applied our proposed approach to the concrete dialog scenario, the performance of the resulting learned strategy was evaluated not only in the simulation domain, but more expressively, within the real world scenario. Hereby we compared the learned strategy's performance to the performance of a non-trivial handcrafted strategy.

1.3 Contents of this work

The presented work is structured as follows: Part two of this thesis will introduce the academic fundamentals upon which this work is build. Subsequently, in chapter three, related work will be presented and its relevancy and applicability with respect to this work will be discussed. Following, in chapter four, we will introduce our proposed procedure model of rapid simulation-driven multimodal strategy learning within human-robot interaction on conceptual level. Subsequently, chapter five will highlight the exemplary application of our approach for our concrete human-robot

² <http://www.sfb588.uni-karlsruhe.de>

dialog scenario. The results of the experiments performed within such practical application will be unveiled and discussed in chapter six. Finally, in chapter seven, a summary and conclusion of this work, as well as suggested further work will be presented.

2 Basics

The following chapter introduces the reader to the basic academic topics and techniques relevant within the scope of this thesis. Hereby, this chapter intends to build up a general ground of expertise, necessary for the further understanding of the presented work. In chapter 2.1, the field of human-machine dialogs will be introduced. Subsequently, within chapter 2.2 we will shift our attention to the formalism of Markov decision processes as well as the technique of reinforcement learning. Following, chapter 2.3 will conclude this chapter by presenting the methodology of reinforcement learning of dialog strategies. Note that within this subchapter additionally an introduction to dialog simulation will be given.

2.1 Human-machine dialogs

Formally, a human-machine dialog is defined as an active interaction between a human user and a computer-aided system, based on sequential turn-taking. The action sphere of both communication partners hereby includes the perception, interpretation, and reaction to the opposite party's activity. Human-machine dialog systems are commonly deployed as an intuitive operation interface for a specific associated application. Hereby, under close collaboration of the dialog system and the application, access to the application's functionality is provided to the human user through natural dialog interaction. Assuming a joint goal of the user and the system, the actions of both communication partners within the dialog interaction can be described as goal-directed and cooperative. The resulting dialogs are then often characterized as task-oriented dialogs.

As a general discrimination, dialog systems are classified according to the used modalities. Dialog systems isolated to the use of speech for interaction are referred to as spoken dialog systems (SDS). Assuming the use of further modalities, such denotation changes to the term of multimodal spoken dialog systems (MSDS). As remarked already earlier, especially in the within this work focused domain of

humanoid robots, multimodal spoken dialog systems receive a high level of attention. Dialog systems developed within this domain are intended to perceive and understand each modality commonly used and relevant within human face-to-face dialog. Pointing gestures, facial expressions, gaze, emotions or general body language hereby only represent a small fraction of imaginable candidates.

In the following, chapter 2.1.1 will provide an overview of different types of dialog applications before we will introduce the general architecture and layers of a spoken dialog system in chapter 2.1.2.

2.1.1 Dialog application types

Human-computer dialogs, nowadays, are employed in very discriminative environments. On the one hand, one can find loquacious chat robots residing in the internet, attempting to entertain its interlocutors from all over the world by providing small talk, telling jokes, and talking in platitudes. Alternatively, on the more business-sensitive side of things, one can find mostly telephone-based dialog systems facilitating for example bank transfers or stock trading via cell phone without any human operator.

In the following section, we will introduce a classification of the most important types of human-computer dialog applications. Hereby we will roughly follow a classification proposed by Pietquin [1].

▪ **Form-filling dialog applications**

Within this type of dialog application, the dialog system aims to achieve a specific amount of pre-defined information slots from the user. Such designed dialogs are also referred to as slot-based dialogs. These applications can be classified as the simplest category of dialog system applications. Usually, the dialog interaction is designed in a straight-forward manner, for example by fix-coding the chronological order of system prompts attempting to fill the existing slots. Optionally, conformational prompts for received pieces of information extend the system's action sphere. Often in such dialog systems, little emphasis is allotted to the linguistic quality of the system prompts or to performing an exceedingly human-like conversation. Goddeau et al. [2] present in their work a typical form filling dialog application, deployed within the domain of used cars advertisements.

▪ **Information retrieval dialog applications**

Most of the dialog systems we are confronted with in our day-to-day life are information retrieval dialog systems. Essentially, hereby the dialog application is connected to an arbitrary knowledge base used to facilitate system responses to performed informational queries by the human user. While these systems are very similar to form-filling systems in the way that they ask for prefixed information attributes, usually a higher degree of system-initiative is incorporated. As an example, within information retrieval dialog systems it is commonly the system's autonomous decision at which point it stops asking for more information and

responds to a particular user query. Such decision-taking is usually performed under the premise of receiving an adequately constrained but still not empty response for a database query. A typical example for an information retrieval dialog application is the "Let's Go" system [3] developed by Raux et al. at Carnegie Mellon University. In this speech- and telephone-based dialog system, users can obtain detailed information about the bus schedule and routes of Pittsburgh's local public transportation system.

▪ **Problem solving dialog applications**

Problem solving dialog applications differ from information retrieval dialog applications in a way that more emphasis is spent on a continuous interaction and strong collaboration between the human user and the system. Typically, problem solving dialogs are employed within the domain of help desks where based on a provided scenario description by the user, the system offers advice for solving a specific problem, e.g. such as how to delete a software virus from a computer workstation. Usually, the system not only suggests the performance of certain actions but also asks the users to report the effects of their execution in order to better interpret the current scenario and adapt the guidance. In a certain sense, hereby the human user takes on the role of an operating assistant of the expert dialog system, which illustrates why a strong collaborative interaction between user and system is essential. An exemplary problem solving dialog application is the system developed by Hipp and Smith [4] which supports users in the repair of an electronic circuit.

▪ **Teaching and tutoring dialog applications**

Similar to problem solving dialog applications, teaching and tutoring dialog applications focus on a collaborative and continuous information exchange between the system and the user. The system's responsibilities hereby can be seen within mediating and teaching certain knowledge in an effective and interactive manner. Such process usually goes along with a continuous control of the user's learning process, in order to detect remaining knowledge gaps of the users and reinitiate a teaching process of the affected issues. Especially in the background of the increasing attractiveness of remote e-learning, teaching and tutoring dialog applications are receiving more and more attention by the research community. The system ITSpoke developed by Litman et al. [5] serves as an example for such dialog application type. Within this system, students are required to solve a set of qualitative physics problems. The system engages in a spoken dialog with the student to provide assistance as well as teach required knowledge in order to ease the problem solving process for the user. Thus, the dialog system ITSpoke can be viewed with respect to its application type as a hybrid system, as it incorporates the aspect of problem solving as well as the aspects of teaching and tutoring.

▪ **Social conversation dialog applications**

As a fifth and final application type we consider social conversation dialog applications. These dialog applications can be generally equated with so called chat-robots, which mostly are deployed within the World Wide Web. In contrast to the

earlier introduced application types, there is no real task-orientation present within such type of dialog systems. Instead, these systems are mainly striving for an authentic simulation of human-like conversational behavior patterns. Hereby the used language model and system actions are often tailored for a dialog context focusing on small-talk topics. As an example, Matsusaka et al. [6] designed such a social conversational dialog application for a robot application. Using a relatively advanced approach, in this work the dialog communication was not restricted to a single user, but instead the robot's purpose was the seamless participation within an ongoing group dialog of human users.

2.1.2 General architecture and layers of a spoken dialog system

Having introduced the different application types of dialog systems, we will now venture forward in the area of dialog systems by addressing their technical realization: now following, the general architecture of spoken dialog systems will be presented. Although within this work we particularly address multimodal dialogs, for this section our focus will be narrowed down to spoken dialog systems.

The modality of speech represents the key communication channel in general human-machine dialogs, as well as in the human-robot interactions we consider within the context of this work. Therefore, the general architecture and layers of a spoken dialog system are worth being introduced in detail. Notice that once a spoken dialog system is set up with its key components, the processing, interpretation and output of additional modalities can usually be incorporated into the spoken dialog system architecture with ease.

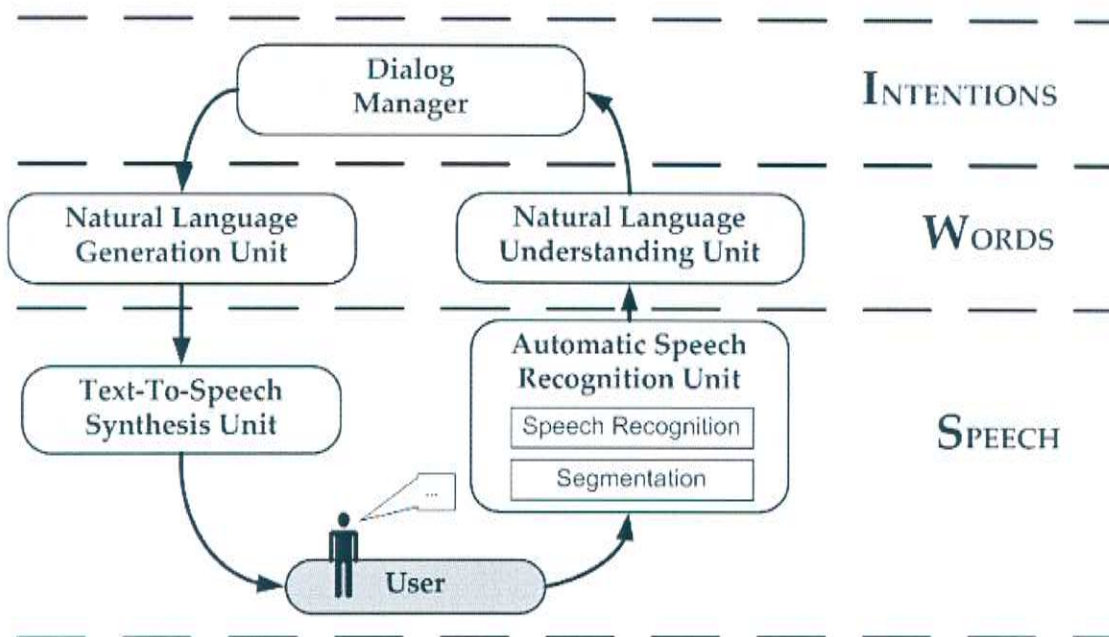


Figure 2.1 General architecture and layers of a spoken dialog system (SDS)

Figure 2.1 shows the general architecture of a spoken dialog system. Within the illustration, five components are demonstrated involved within the processing, interpretation and reaction to a user utterance. The components' associated levels of processing are stated on the right side of the illustration. In the following, a brief and cursory introduction will be given to each component and its responsibility.

2.1.2.1 Automatic speech recognition unit

The first component to become active after an occurred user utterance is the automatic speech recognition (ASR) unit. Working directly on the speech signal of the user, the task of the ASR unit is to transcribe the input speech signal into a textual form. Obviously, such processing requires the performance of multiple steps which will not be discussed in detail within the scope of this thesis. A detailed presentation, however, can be found in [7] and [8]. Generally, the process of automatic speech recognition is carried out in a way that an initial segmentation subunit first identifies speech segments - so called utterances - within the incoming continuous acoustic signal. Such segmentation can be performed for example by using an acoustic signal activity automaton. In a second step, feature extraction from the speech signal is performed on an utterance level. Hereby several transformations are applied to the speech signal, reducing the signal's redundancy as well as facilitating the extraction of a finite set of feature vectors. The design of such feature vectors is targeted to optimally extract all valuable information from the speech signal in a compressed and easily processable manner. Once feature extraction is performed, statistical methods are applied. An acoustic model provides a probabilistic mapping of state features to its corresponding acoustic unit. The granularity of such an acoustic unit is ultimately system-dependant and is usually represented by phonemes or even finer sub-phonemes. Based on such acoustic representation, a trained language model - usually realized by a Hidden Markov Model (HMM) - then finally allows the estimation of a word sequence which best matches the original acoustic speech signal. Hereby, usually ASR systems not only return the one most likely solution, but instead a n-best hypotheses list, each word sequence having a confidence level assigned determining the list's order.

2.1.2.2 Natural language understanding unit

Receiving one or multiple estimated word sequences from the ASR unit, the natural language understanding (NLU) unit aims to extract the overall semantic meaning of the word sequence. Thereto, typically three different phases are traversed in order to facilitate this intended semantic interpretation. In the first step, a syntactical analysis of the word sequence is performed. Such process can be helpful for example in order to dissolve potential ambiguities within the word sequence. Secondly, the context-independent semantic information encompassed within the word sequence is analyzed. In doing so, ontologies³ have become a widely used and popular mean

³ Roughly spoken, ontologies allow the classification of words or group of words in a subset of synonyms which are defined within the system's world knowledge and therefore consistently interpretable by the system.

within the domain of semantic interpretation. In the third step, the extracted semantic information is evaluated and potentially refined in the direct dialog context. Frequently, only by considering the discourse of a dialog, i.e. the historic utterances within a dialog interaction, the semantic meaning of a word sequence can be determined correctly. For example, a user's utterance of "Yes" is completely meaningless in terms of its semantic information when neglecting the contextual information. Incorporating the preceding system action within the interpretation process, however, allows determining the context-sensitive and thereby meaningful semantic representation of the user utterance.

2.1.2.3 Dialog manager

The dialog manager component represents the nerve and operation center and of any dialog system. Dependent on the application domain, the dialog manager stays in close communication with all for the dialog system relevant operation units, in order to facilitate immediate system responses of any kind to occurred dialog interactions. For example, within a human-robot dialog interaction, a short and effective communication path between the dialog manager and the robot control system is required, in order to facilitate a, with respect to the dialog interactions, consistent robot behavior. Furthermore, the dialog manager component is in charge of maintaining and adapting the current dialog state, including discourse management. Note that hereby the dialog manager component works usually not on a word level, but instead on an intentional level. Essentially such intentions are represented by the informational concepts a user wants to communicate (also referred to as speech acts or dialog acts within dialog theory). In this context, depending on the processing and analysis of the received information from the NLU unit and its underlying intentions, the dialog manager carries out a particular dialog strategy which determines the next system action for the current dialog state. Such system actions are typically speech outputs triggered through the natural language generation unit, but might also include system-internal activities, such as the adaptation of the language model. Generally, the realization of such action selection process can vary from the use of deterministic rule-based approaches to completely probabilistic action selection. Finally, the dialog manager component also assumes an important role within the domain of a multimodal dialog design: Often such component is in charge of the coordination and fusion of multimodal user inputs.

2.1.2.4 Natural language generation unit

Assuming the dialog manager picked a speech output as the next system action to be performed, it is the natural language generation (NLG) unit's responsibility to transform such action into natural language. While NLG units of some dialog systems can comprise a high degree of complexity (e.g. by performing text generation dynamically on world-level from abstract action definitions), for the majority of spoken dialog systems a more simplistic approach suffices: the definition and use of fixed system prompts for particular system actions. Especially for slot-based information retrieval dialog systems, the fixed definition of system prompts for each existing information slot in addition to answer, greeting and farewell

statements is common use. In order to retain some degree of flexibility, placeholders are frequently utilized within the system prompts. Then, the dialog system becomes capable of explicitly confirming prior received information but also is able to personalize the dialog in an enhanced way, e.g. by incorporating the eventually known name of the system user within the dialog.

2.1.2.5 Text-To-Speech synthesis unit

Once a textual utterance is generated by the NLG unit, the application of a text-to-speech (TTS) synthesis unit allows its transformation into a synthetic acoustic output signal. Compared to the use of human voice recordings, synthetic speech generation is known for its artificial character, however, also offers a high degree of agility to the developer as principally any desired textual information can be expressed dynamically during runtime. Modern TTS systems nowadays are mostly using the so called concatenative synthesis method for speech generation. Hereby a huge database of prior recorded segments of human speech serves to compose and synthesize any arbitrary word sequence. Depending on the sophistication of the TTS system, the recorded segments are available on different acoustic levels of speech, e.g. in form of phones, syllables, morphemes, words, or complete phrases. Additionally, often also prosodic features are utilized in order to improve the naturalness of the speech output. With the output of the system prompt by the TTS system the processing cycle of a user utterance closes and the system is ready to accept the next user action.

2.2 Markov decision processes and reinforcement learning

This subchapter will give an introduction to Markov decision processes (MDP) as well as the basics of reinforcement learning. In its core, reinforcement learning can be understood as the application of unsupervised learning algorithms, which facilitate the determination of optimal behavior strategies for so called Markov decision processes. Correspondingly, in the following, first the formalism of Markov decision processes will be introduced. Based on such knowledge, then the field of reinforcement learning and its techniques will be presented.

2.2.1 Markov decision processes

Markov decision processes represent a sophistication of the more general, so called sequential decision processes (SDP). While the information that makes a SDP a MDP will be withheld from the reader for the moment, we first want to focus our attention on the nature of sequential decision processes. For such processes, we assume the existence of an agent which repeatedly is taking actions at discrete points of time, based on the observed current state of its environment. Hereby, every action of the agent triggers a numerical reward which indicates the adequateness of the agent's last action decision. Figure 2.2 illustrates the SDP formalism.

All sequential decision processes are composed by the definition of the subsequently described tuple of the state space S , the action set A , the state transition function T , as well as the reward function R .

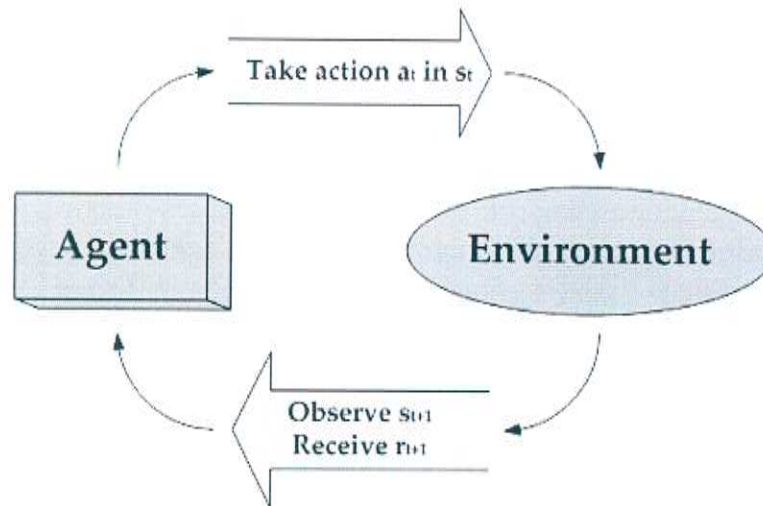


Figure 2.2 Sequential decision processes

▪ S: State space

The state space S refers to the set of states reachable for the agent within the SDP. Essentially, the state information corresponds to the agent's observation of the environment and usually is determined by the current status of a finite number of defined state variables. The particular state an agent is visiting at time t is denoted by s_t .

▪ A: Action set

The action set A contains all actions available to the agent within the complete state space. Depending on the current state the agent is visiting, however, the set of executable action choices for the agent may be restricted to a subset of A , as some actions might be only applicable given a particular environmental condition. The action the agent takes at time t is indicated as a_t .

▪ T: State transition function

The state transition function T describes the dynamics of the agent's interaction with the environment. Given that the agent takes action a_t in state s_t , T provides a probability distribution in which successor state, that is s_{t+1} , the agent will arrive. Hereby the probability P for observing successor state s_{t+1} after the execution of action a_t in state s_t is specified within an SDP by $P(s_{t+1} | s_t, s_{t-1}, \dots, s_0, a_t, a_{t-1}, \dots, a_0)$ and thereby based on the history of all previously visited states and actions taken by the agent.

▪ R: Reward function

As a consequence of every action a_t the agent takes, a numerical reward r_{t+1} is generated. The reward function R precisely specifies a probabilistic distribution of this generated reward. Again, based on the history of all previously visited states and actions taken by the agent, the probability for receiving reward r_{t+1} after the execution of action a_t in state s_t is specified within an SDS by $P(r_{t+1} | s_t, s_{t-1}, \dots, s_0, a_t, a_{t-1}, \dots, a_0)$.

As mentioned earlier, Markov decision processes represent a specialization of sequential decision processes. Hereby it is one particular condition with respect to the state-space configuration which is specifying if a SDP is also considered a valid MDP: Such necessary condition for a MDP is the compliance of its state signal with the so called Markov property. The Markov property holds when the conditional probability distribution of a future state depends only upon the preceding state and the therein taken action. In other words, the Markov property demands that every piece of information influencing the probability of particular state transitions has to be encoded within the state signal. Formally, every valid MDP fulfills the following Markovian property regarding the state transition function T :

$$(2.1) \quad T_{ss'}^a = P(s_{t+1} = s' | s_t = s, a_t = a) = P(s_{t+1} | s_t, s_{t-1}, \dots, s_0, a_t, a_{t-1}, \dots, a_0)$$

Given the Markov property, an optimal policy π , i.e. taking always optimal decisions regarding which actions to take in which states, can be automatically computed for a particular MDP. Precisely this benefit is the reason for the popularity and high potential of the MDP formalism. The algorithms, able to determine an optimal policy within an MDP, are what we refer to as reinforcement learning algorithms and will be presented now in the following subchapter.

2.2.2 Reinforcement learning

This subchapter will provide a basic introduction to the field of reinforcement learning. In this context, the nature and origin of reinforcement learning, along with the various available methods for reinforcement learning, will be discussed.

2.2.2.1 Introduction

Reinforcement learning as proposed by Sutton and Barto [9] is a computational approach to the process of learning from interaction and experience. Within each human life, and also that in the animal kingdom, we find numerous examples of learning behaviors which resemble significantly with what we understand as reinforcement learning. An infant that touches a hot stove for the first time will experience the negative consequence of burning pain. It will consequently be stigmatized from this experience, and will attempt to avoid the same situation in the future. On the other hand, a dog receiving a delicious tidbit every time it follows a

command from his master, will soon be able to associate the positive reward with its preceding action and will start to behave correspondingly. In the literature, such learning behavior is also referred to often as trial-and-error learning. The interesting phenomenon of these learning processes is that they occur in an unsupervised manner. There is no particular teacher in the picture demonstrating the correct behavior in a supervised way. This is exactly the type of learning addressed in typical reinforcement learning tasks. Correspondingly, reinforcement learning is also denoted as an unsupervised learning method, and thereby takes a special position among the collection of mostly supervised machine learning techniques.

Learning in an unsupervised manner principally facilitates the convenience of placing our so called learning agent in a completely unknown environment. However, by interaction with its environment, the agent is able to learn step by step how to act in an adequate way, just by observing the consequences of its actions in the form of received rewards. Hereby, obviously the kind of feedback we introduced in our earlier examples, like severe pain or delicious food, is not appropriate and processable in the computational domain. Instead, reinforcement learning uses the formalism of numerical rewards to model positive and negative feedback. The maximization of such rewards then represents optimal behavior and represents the overall goal of reinforcement learning.

Two central issues arise in any application of reinforcement learning which can be regarded as particularly challenging. Firstly, there is the challenge of a fair trade-off between exploration and exploitation. In order to learn about the dynamics and potential of its environment, it is necessary that the learning agent profoundly explores all possible actions and states. Otherwise, the agent could never be sure about making the best choices, as unexamined actions, and therefore undiscovered states, could comprise maximal but unknown benefits. However, in the course of optimizing its behavior, the agent also has an urge of exploiting its so far collected knowledge about the environment by preferably taking actions that caused promising rewards in the past. The importance of a healthy balance between exploration and exploitation becomes obvious, particularly when assuming a dynamically changing environment. For such given condition, rewards for certain actions in specific states might change over time. Correspondingly, following exclusively an exploitation strategy after a short learning period bares the risk of ignoring the environmental changes which could result in increasingly sub-optimal behavior. On the other side, the excessive use of exploration moves instead of experience-based action selection will presumably cause the loss of valuable reward. As we will see later in this chapter, reinforcement learning techniques address this issue in different ways.

A second challenging issue in reinforcement learning tasks is the decision of how to weigh intermediate reward in relation to delayed rewards. As an example, a student preparing for an exam might consider the daily study sessions in correlation with the resulting lack of free time as a negative intermediate reward. However, receiving an excellent grade in the exam at a later point of time would represent a positive

delayed reward, which might compensate the time-consuming learning effort in the first place. Depending on how short- or farsighted the student's attitude is, i.e. how valuable rewards in the future are considered, but also how important the student considers a good grade in comparison to his available free time, the student's optimal behavior - represented in this particular case by his learning effort - varies. Positively, reinforcement learning techniques are capable of successfully handling the concept of delayed rewards.

2.2.2.2 Value Functions

In order to find optimal policies within an MDP, all reinforcement learning techniques fall back on one central instrument, so called value functions. Two different types of value functions exist within reinforcement learning: state value functions, usually denoted by V^π , and action-state value functions, abbreviated as Q^π . The state value $V^\pi(s)$ specifies an estimate of the upcoming value potential of a particular state s when following policy π . Note that in the context of reinforcement learning, the understanding of value can be equated with the understanding of expected reward. More precise, the associated V value of a state s expresses how much future reward can be expected when visiting state s and subsequently following policy π . Formally written, $V^\pi(s)$ can be defined for any MDP as

$$(2.2) \quad V^\pi(s) = E_\pi\{R_t | s_t = s\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right\},$$

where E_π represents the expected return and γ a discount factor between zero and one, determining the weight of rewards in the future. Note that every terminal state, i.e. no further action is applicable in that state within the MDP, automatically owns a state value of zero.

The second type of value function mentioned earlier, the action-state value function, extends the introduced state value function in a way that it not only computes the expected return for every single state of an MDP, but for every valid action-state pair present within the MDP. Correspondingly $Q^\pi(s, a)$ defines the expected return of taking action a when visiting state s . Similarly to the definition above, the action-state value function $Q^\pi(s, a)$ is formally defined as

$$(2.3) \quad Q^\pi(s, a) = E_\pi\{R_t | s_t = s, a_t = a\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right\}$$

The attentive reader might realize already why it might be more promising to compute an action-state value function than computing a pure state value function. Learning the optimal policy in an MDP corresponds to learning to take the optimal action in every reachable state. While a state value function can only give indirect hints as to which actions seem promising in a particular state s , an action-state value

function allows us to directly compare the potential of all actions applicable for a specific state in order to then pick the action with the maximal action-state value. And yet, the fact that the space of the possible action-state pairs is significantly bigger than the space of ordinary states requires mentioning. Consequently, computing the action-state value function will always cost more computational effort than learning the state value function.

When taking a closer look at the above equations given for the state value function and action-state value function, one might discover an essential and fortunate characteristic such value functions hold: the tight correlation of a state value to the value of its potential successor states facilitate a recursive formulation of the value function, also called the Bellman equation. In case of the value function $V^\pi(s)$, the corresponding Bellman equation is derived as followed:

$$\begin{aligned}
 V^\pi(s) &= E_\pi\{R_t | s_t = s\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right\} \\
 &= E_\pi\left\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s\right\} \\
 (2.4) \quad &= \sum_a \pi(s, a) \sum_{s'} T_{ss'}^a [R_{ss'}^a + \gamma E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_{t+1} = s'\right\}] \\
 &= \sum_a \pi(s, a) \sum_{s'} T_{ss'}^a [R_{ss'}^a + \gamma V^\pi(s')]
 \end{aligned}$$

where \sum_a represents the set of applicable actions in state s , $\sum_{s'}$ the set of potential successor states of s , and $\pi(s, a)$ the probability of executing action a in state s given a stochastic policy π . For the action-state value function $Q^\pi(s, a)$, the Bellman equation can similarly be evolved as followed:

$$\begin{aligned}
 Q^\pi(s, a) &= E_\pi\{R_t | s_t = s, a_t = a\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right\} \\
 (2.5) \quad &= E_\pi\left\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s, a_t = a\right\} \\
 &= \sum_{s'} T_{ss'}^a [R_{ss'}^a + \gamma E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_{t+1} = s'\right\}] \\
 &= \sum_{s'} T_{ss'}^a [R_{ss'}^a + \gamma Q^\pi(s', a)]
 \end{aligned}$$

As we will see in the following, the Bellman equations have a fundamental significance for reinforcement learning methods, as they provide a suitable basis for different approaches to calculate or respectively approximate $V^\pi(s)$ and $Q^\pi(s, a)$.

2.2.2.3 Reinforcement learning methods

Within the domain of reinforcement learning, different methods exist for finding the optimal policy within a reinforcement learning task. Barto and Sutton [9] propose a categorization of reinforcement learning methods into three elementary classes: dynamic programming, Monte Carlo methods, and temporal-difference learning. While all of these methods are designed to be applied to the formalism of MDPs (see chapter 2.2.1), their individual characteristics differ significantly which ultimately results in different runtime and convergence behavior. The functioning and differences of these methods will be now discussed in detail.

2.2.2.3.1 Dynamic programming

Dynamic programming is the most efficient and accurate method for solving reinforcement learning problems. However, as a key prerequisite, applying dynamic programming to a reinforcement learning task assumes a perfect model of the environment. In the terms of a MDP, a perfect model of the environment is represented by the full and accurate knowledge of the state transition function T as well the reward function R . However, in practice, the presence of such perfect environmental model within typical reinforcement learning tasks is rather rare. Therefore, dynamic programming can be only rarely applied to real reinforcement learning tasks. Nevertheless, the idea of dynamic programming is elementary to the field of reinforcement learning from a theoretical point of view. As we will see later on, the more relaxed Monte Carlo methods as well as temporal-difference learning principally try to imitate the behavior of dynamic programming methods, but without requiring a perfect model of the environment.

In chapter 2.2.2.2, the Bellman equations for $V^\pi(s)$ and $Q^\pi(s, a)$ were introduced. At that point, we were looking at policy π as any policy available in the MDP. Obviously, there is one particular policy within the set of policies we are especially interested in when considering the domain of reinforcement learning: the optimal policy π^* . However, in order to follow the optimal policy, it is necessary to know its corresponding value function. We will refer to these particular value functions in the following as $V^*(s)$ and $Q^*(s, a)$ with

$$\begin{aligned}
 (2.6) \quad V^*(s) &= \max_a E\{r_{t+1} + \gamma V^*(s_{t+1}) \mid s_t = s, a_t = a\} \\
 &= \max_a \sum_{s'} T_{ss'}^a [R_{ss'}^a + \gamma V^*(s')]
 \end{aligned}$$

and

$$\begin{aligned}
 (2.7) \quad Q^*(s, a) &= E\{r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') \mid s_t = s, a_t = a\} \\
 &= \sum_{s'} T_{ss'}^a [R_{ss'}^a + \gamma \max_{a'} Q^*(s', a')]
 \end{aligned}$$

where s' is the successor state of s and a' the associated taken action in state s' . As we can see, the above equations are directly derived from the Bellman equations, but focus on an orientation towards the optimization of both value functions, indicated by the max terms. Due to their origin from the Bellman equations, the above formulas are also called Bellman's optimality equations.

As mentioned earlier, when applying dynamic programming to a reinforcement learning task, we assume a perfect model of the environment, i.e. that $T_{ss'}^a$ and $R_{ss'}^a$ are known. Hereby, in order to compute the state value function of an arbitrary policy $V^\pi(s)$, a system of $|S|$ (where S is the state space) linear equations with $|S|$ unknowns (the $V^\pi(s)$ values) had to be solved. As the computation of the exact solution of such a system can be very expensive (especially when assuming a large state space), dynamic programming algorithms fall back on iterative solutions. Using the Bellman equation for $V^\pi(s)$ as an update rule in the way as shown in equation 2.8, it is possible to closely approximate the exact solution of the value function for an arbitrary policy π by running repeatedly so called full backups (i.e. every state value is updated once), until the state values converge. This process is also referred to as policy evaluation.

$$(2.8) \quad V_{k+1}(s) = \sum_a \pi(s, a) \sum_{s'} T_{ss'}^a [R_{ss'}^a + \gamma V_k(s')]$$

Given the instrument of policy evaluation, we are able to initiate our search for the optimal policy π^* . We start off by determining an arbitrary initial policy π_0 and compute its value function $V^{\pi_0}(s)$ through policy evaluation as shown above. Once the state value function is found, we can easily compute $Q^\pi(s, a)$ using equation 2.7. The knowledge about $Q^\pi(s, a)$ allows us then to investigate our policy π_0 more closely. Assuming that our policy π_0 is optimal for every state in $|S|$ and its applicable actions, the following condition would hold:

$$(2.9) \quad Q^{\pi_0}(s, a) \leq V^{\pi_0}(s).$$

However, a violation of the above condition for a single action-state pair (s, a) would mean that taking such particular action a in state s - while following policy π_0 thereafter - would promise a higher expected reward than following strictly policy π_0 in state s . Clearly, given such case, the evaluated policy is not yet optimal but instead we would assume policy π_{n+1} to be more close to the optimum with

$$(2.10) \quad \pi_{n+1}(s) = \arg \max_a Q^{\pi_n}(s, a).$$

The process of refining π_n towards π_{n+1} is called policy improvement. By repeatedly performing the processes of policy evaluation and policy improvement in a change, it is now possible to steadily improve our policy until finally condition 2.9 will hold

for every state. In that moment, our algorithm reached our overall goal with the assured optimality of our strategy ($\pi_n = \pi^*$) and the validity of equation 2.11 for every state s defined within the MDP.

$$(2.11) \quad \max_a Q^{\pi_n}(s, a) = V^{\pi_n}(s)$$

The overall process of continuously tuning an initially arbitrary policy π_0 to the optimal strategy π^* is referred to as policy iteration. Listing 2.1 shows such just explained mechanism of policy iteration using dynamic programming in pseudo code.

```

Step 1: Initialization
     $V(s) \in \mathbb{R}$  and  $\pi(s) \in A(s)$  arbitrarily for all  $s \in S$ 

Step 2: Policy Evaluation
    Repeat
         $\Delta \leftarrow 0$ 
        For each  $s \in S$  :
             $v \leftarrow V(s)$ 
             $V(s) \leftarrow \sum_{s'} T_{ss'}^{\pi(s)} [R_{ss'}^{\pi(s)} + \gamma V(s')]$ 
             $\Delta \leftarrow \max(\Delta, |v - V(s)|)$ 
        Until  $\Delta < \theta$  (where  $\theta$  is a small positive threshold)

Step 3: Policy Improvement
    policy-stable  $\leftarrow$  true
    For each  $s \in S$  :
         $b \leftarrow \pi(s)$ 
         $\pi(s) \leftarrow \arg \max_a \sum_{s'} T_{ss'}^a [R_{ss'}^a + \gamma V(s')]$ 
        If  $b \neq \pi(s)$ , then policy-stable  $\leftarrow$  false
    If policy-stable, then end; else go to Step 2
  
```

Listing 2.1 Pseudo code for policy iteration (dynamic programming)

2.2.2.3.2 Monte Carlo methods

As shown above, dynamic programming represents an effective way of computing the optimal policy within an MDP. Note that when using dynamic programming, the optimal policy is not obtained through a real learning process, but more through a systematic computational procedure. This changes, however, when we look at Monte Carlo methods. Unlike dynamic programming, Monte Carlo methods do not require a perfect model of the environment, which ultimately gives them a more pragmatic and popular role within reinforcement learning. Instead, Monte Carlo methods literally learn from experience, i.e. from observed trajectories of states, actions and rewards, experienced by sample interactions with the environment. Such trajectories might be studied from real interactions (so called online experience) or from

simulated interactions. Clearly, using a simulation-driven approach leads to some extent again to the mentioned pitfall of dynamic programming methods: in order to be able to generate simulation interactions, moderate knowledge about the environment has to be available. In many cases, however, it shows to be relatively straight-forward to simulate experience for a given reinforcement learning task, while it is impossible to explicitly determine the state value function T and reward function R . Even more convenient than building a simulation model, the exclusive use of online experience would save the effort of building any kind of environmental model when using Monte Carlo methods. However, the acquisition of online experience in a great extent, as necessary for Monte Carlo methods, usually is associated with high costs and thereby often infeasible.

Similar to dynamic programming, Monte Carlo methods also use the technique of policy evaluation and policy improvement to find the optimal policy within a given MDP. However, as Monte Carlo methods only work on basis of sampled experience, the computation of the value function V^π differs significantly. As laid out before, $V^\pi(s)$ represents the future expected reward of a state particular state s when following policy π . Correspondingly, it is evident that a valid approach to approximate $V^\pi(s)$ would be to look at all rewards observed after the visit of a state s within the available experience, and to average such observed rewards over the number of interactions. Such mean value would obviously represent a valid estimation of the state value of state s . Hereby it is important that all evaluated experience was collected exclusively while following policy π . We then would expect that with time the average values for every state would converge to $V^\pi(s)$. It is precisely such approach of computing $V^\pi(s)$ that forms the core of Monte Carlo methods. Formally, the state value function and action state value function are specified for Monte Carlo methods as follows:

$$(2.12) \quad V^\pi(s) = \text{average}(\text{Rewards}(s))$$

$$(2.13) \quad Q^\pi(s, a) = \text{average}(\text{Rewards}(s, a))$$

A big benefit of Monte Carlo methods is that they facilitate the learning of the action-state value function $Q^\pi(s, a)$ directly from experience. Listing 2.2 shows a simple Monte Carlo algorithm in pseudo code, hereby assuming the availability of a sufficient amount of experience. As we can see, for every experience the visited -state values $Q^\pi(s, a)$ are updated correspondingly, and afterwards policy π is improved towards the optimal strategy. We assume policy π to represent the optimal policy as soon as a sufficiently defined number of interactions occurred without causing any changes to policy π .

It can be shown that Monte Carlo methods are guaranteed to converge towards the same optimal value function as dynamic programming methods, assuming that every action-state pair is visited an infinite number of times. A point of criticism for Monte Carlo methods is that updates to the action-state values and policy π are only

made at the end of every experienced episode. This property can evolve into a major drawback, especially in reinforcement learning scenarios with long interaction episodes. However, as we will see in the next subchapter, the method of temporal-difference learning precisely addresses and adequately solves this obstacle.

```

Initialize for all  $s \in S, a \in A(s)$  :
   $Q(s, a) \leftarrow$  arbitrary
   $\pi \leftarrow$  arbitrary
  Rewards( $s, a$ )  $\leftarrow$  empty list

Repeat until  $\pi$  is stable:
  (a) Generate an experience  $e$  using  $\pi$ 
  (b) For each pair ( $s, a$ ) appearing in experience  $e$ :
       $R \leftarrow$  reward following the first occurrence of  $s, a$ 
      Append  $R$  to Rewards( $s, a$ )
       $Q(s, a) \leftarrow$  average(Rewards( $s, a$ ))
  (c) For each  $s$  in the episode:
       $\pi(s) \leftarrow \arg \max_a Q(s, a)$ 

```

Listing 2.2 Pseudo code Monte Carlo algorithm

2.2.2.3.3 Temporal-difference learning

The most popular and applied reinforcement learning technique so far, without any doubt, is temporal-difference (TD) learning. In comparison to Monte Carlo methods, the big advantage of TD methods is their feature of performing updates of state values based on current estimates of other state values, also referred to as bootstrapping mechanism. While Monte Carlo methods insist upon waiting until the end of an episode to perform state value updates, TD methods facilitate online updates of state values in the time the immediate reward r_{t+1} of an action a_t is observed. The update rule of the simplest TD method, TD(0), can be formulated for the state value function V as follows

$$(2.14) \quad V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$$

where α represents the learning factor, usually chosen somewhere around 0.1. While learning state value estimates from other approximations might seem risky and unstable at first sight, it can be shown that for any fixed policy π , using the update rule above guarantees a convergence of the state value function to V^π . Requirement for such guarantee of convergence, however, is a well selected decrease function towards a sufficient small learning factor α .

As pointed out earlier, even more promising than learning the state value function V is learning the action-state value function Q for a particular policy, as it allows the direct acquisition of the optimal policy. For TD learning, we distinguish such action-state based methods in on-policy learning and off-policy learning.

On-policy learning implies learning the action-state value of a policy that is followed during experience acquisition (the so called behavior policy). For temporal-difference learning, such learning method is also referred to as the SARSA algorithm, referring to the fact that its update formula requires the quintuple of $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$. Listing 2.3 shows the pseudo code for a SARSA algorithm. Using an ε -greedy policy, i.e. the so far learned optimal action is taken with probability $(1-\varepsilon)$ while exploration moves are executed with probability ε , it is possible to estimate Q^π for such behavior policy.

```

Initialize  $Q(s, a)$  arbitrarily

Repeat (for each episode):
  Initialize  $s$ 
  Choose  $a$  from  $s$  using policy derived from  $Q$  (e.g.  $\varepsilon$ -greedy)
  Repeat (for each step of episode)
    Take action  $a$ , observe  $r, s'$ 
    Choose  $a'$  from  $s'$  using policy derived from  $Q$  (e.g.  $\varepsilon$ -greedy)
     $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$ 
     $s \leftarrow s'; a \leftarrow a'$ 
  Until  $s$  is terminal

```

Listing 2.3 Pseudo code for the SARSA algorithm

Off-policy learning, on the other hand, introduces an interesting new aspect of reinforcement learning: learning about one policy, while following another. In this context, Watkins proposed in 1989 the so called Q-learning algorithm [10], which until today is seen as one of the major breakthroughs within reinforcement learning. The specialty of this algorithm is based on the fact that - independent from the policy followed - the learned action-state value function Q directly approximates the action-state value Q^* of the optimal policy π^* . The update rule for an action-state value is formulated as followed for Q-learning:

$$(2.15) \quad Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

It can be shown that Q-learning converges with probability one to Q^* , given the following two conditions: Firstly, the followed policy continuously visits and thereby updates all action-state pairs. And secondly, the learning parameter α is decreased over time to a sufficiently small value. Again, the extraordinary aspect of this method is that while respecting the above properties, any policy of choice could be followed for experience generation and still the optimal policy would be learned. This feature makes Q-learning tremendously popular and pleasant in practice. Listing 2.4 shows the Q-learning algorithm in pseudo-code.


```

Initialize  $Q(s, a)$  arbitrarily

Repeat (for each episode):
  Initialize  $s$ 
  Repeat (for each step of episode):
    Choose  $a$  from  $s$  using policy derived from  $Q$  (e.g.  $\epsilon$ -greedy)
    Take action  $a$ , observe  $r, s'$ 
     $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$ 
     $s \leftarrow s'$ 
  Until  $s$  is terminal

```

Listing 2.4 Pseudo code for Q-learning

2.2.2.3.4 Eligibility traces

As a concluding point in this preamble to reinforcement learning, we want to introduce the idea of eligibility traces in reinforcement learning, which can be well combined with e.g. the Q-learning algorithm. Eligibility traces can be considered a hybrid approach of Monte Carlo methods and TD learning. In essence, action-state values are updated neither based only on the last visited action-state value estimate nor on the observed rewards of a whole episode, but by a mixture of both. Precisely the factor λ , chosen between zero and one, determines within which balance concrete observed rewards and other state value estimates are incorporated into the value update process. While setting the λ value to zero can be seen equivalent to TD(0) learning, a λ value of one refers to typical Monte Carlo methods. In this context, Watkins' $Q(\lambda)$ method [10] has become popular due to its easy implementation and fair convergence performance and will now be discussed further.

Within Watkins' $Q(\lambda)$ method, every action-state value owns an eligibility trace value $e(s, a)$. Every time a particular action-state pair is visited, the eligibility trace is set to one for this action-state pair. Additionally, the eligibility traces of action-state pairs sharing the same state with the visited action-state pair are set to zero. All earlier increased other action-state eligibility traces are discounted by a discount factor γ and the earlier described λ value. Equation 2.16 shows the corresponding equation for the eligibility traces.

$$(2.16) \quad e_t(s, a) = \begin{cases} 1 & s = s_t, a = a_t \\ 0 & s = s_t, a \neq a_t \\ \gamma \lambda e_{t-1}(s, a) & s \neq s_t \end{cases}$$

Updates of all Q-values are then performed at every step according to the following update rule:

$$(2.17) \quad Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)) \cdot e_t(s, a)$$

One important issue that has to be considered when using Watkins' $Q(\lambda)$ method is that potential exploratory moves are excluded from the learning process. Learning the optimal action-state values requires learning from exclusively greedy actions. Correspondingly, once the behavior policy performed a non-greedy exploratory action, it becomes necessary to reset all eligibility traces to zero. Only with such a proceeding it can be assured that non-optimal, explorative moves are excluded from the learning process. Listing 2.5 shows the pseudo code of Watkins' $Q(\lambda)$ method.

```

Initialize  $Q(s, a)$  arbitrarily and set  $e(s, a) = 0$  for all  $s, a$ 

Repeat (for each episode):
  Initialize  $s, a$ 
  Repeat (for each step of episode)
    Take action  $a$ , observe  $r, s'$ 
    Choose  $a'$  from  $s'$  using policy derived from  $Q$  (e.g.  $\epsilon$ -greedy)
     $a^* \leftarrow \arg \max_b Q(s', b)$ 
     $\delta \leftarrow r + \gamma Q(s', a^*) - Q(s, a)$ 
     $e(s, a) \leftarrow 1$ 
    For all  $s, a$ 
       $Q(s, a) \leftarrow Q(s, a) + \alpha \delta e(s, a)$ 
      If  $a' = a^*$ , then  $e(s, a) \leftarrow \gamma \lambda e(s, a)$ 
      else  $e(s, a) \leftarrow 0$ 
     $s \leftarrow s'; a \leftarrow a'$ 
  Until  $s$  is terminal

```

Listing 2.5 Pseudo code for Watkins' $Q(\lambda)$ method

The use of eligibility traces within reinforcement learning is especially beneficial given the presence of a extensive action-state space. In such scenarios, propagating rewards to more than the last visited state facilitates faster and more frequent updates of a wider scope of action-state values compared to simple Q-learning. However, due to its performed simultaneous updates of several action-state values, the use of eligibility traces also endangers a stable strategy learning process. Hereby, an adequate specification of the λ value plays a crucial role for a moderate performance of Watkins' $Q(\lambda)$ method and often is observed to be highly dependent on the given action-state space configuration.

2.3 Reinforcement learning of dialog strategies

We will now introduce the methodology of applying reinforcement learning within the domain of dialog systems, aiming for the computation of optimal dialog strategies. Firstly, generalizations about the necessary formulation of dialog as a Markov decision process will be established. Subsequently, the two primary, yet different, approaches to strategy learning will be introduced.

2.3.1 Formulating dialog as a Markov decision process

In order to be able to apply reinforcement learning to dialog systems, a basic prerequisite is the formulation of the dialog system as a Markov decision process. Recall that chapter 2.2.1 gave a brief introduction to Markov decision processes, introducing the main elements of such formalism as the state space S , the action space A , the state transition function T , as well as the reward function R . It is the system designer's responsibility to design the Markov decision process in a way that it adequately represents the dialog system's dynamics and its environment. The work of Levin et al. [11] in 1998 pioneered such transformation for real dialog systems. In the following, we will introduce some generalities about the design of the state space, action set, and reward function within the mapping of a dialog system to a Markov decision process.

- **State space**

For the state space configuration of the dialog system, special care has to be taken for a moderate compliance with the Markov property, i.e. that the state and reward at time $t+1$ only depend on the state and action at time t . Only assuming such property of the defined state space, the learned dialog strategy can be considered fully optimal. Correspondingly, the chosen state features should cover all essential dialog state information influencing the dialog dynamics. Considering the exponential growth of the state space with every state variable, an adequate design of such state features and their individual encoding can be an arduous task. Within this course, a tradeoff often needs to be established: while few features and thereby a small state space on the one side facilitates fast policy learning, more state features and thereby a bigger state presumably will facilitate learning a more sophisticated strategy. In general, the state-space design is very task-dependant and, therefore, no generic rules can be given for a proper selection of state features. As one example, for information slot based dialogs, typically the current status of the information slots (e.g. empty, filled or confirmed) are chosen as elementary state features.

- **Action set**

Compared to the state space configuration, the design of the action set within the MDP is usually a trivial task as such action space commonly equates with the available system actions of the dialog system. Generally, there are two types of actions that are distinguished: actions that are directed towards the user and those that are system-internal actions. In informational retrieval dialog systems, actions directed toward the user are usually represented by the system uttering a particular social phrase (e.g. greeting/closing statements), informational phrase (e.g. "Your desired train leaves today at 1:12pm"), question prompts (e.g. "Where do you want to travel?"), or conformational requests (e.g. "Did you say you want to travel tomorrow?"). Action choices for the system can be present by the availability of different initiatives of system prompts: options hereby include the use of system-initiative, i.e. the user is prompted by a closed question for a particular information ("Do you want to perform a credit transfer?"), or mixed-initiative, i.e. asking an

opened question and leaving the initiative of providing information to the user (e.g. “How can I help you?”). Further, for conformational requests, available action choices may consist of using explicit confirmation (e.g. “Did you say you are living in the US?”) or implicit confirmation (e.g. “In which state in the US are you living?”). System-internal actions on the other side can not be classified as strictly. In typical information retrieval dialog systems they would be represented by database queries or access of external knowledge bases. But also more dialog system specific actions are imaginable, as for example an action that triggers an alteration of the language model, depending on the current dialog state.

▪ Reward function

The design of the reward function for a particular dialog scenario is a crucial point for the quality of the learned strategy, as it presents the central optimization criterion. As a first step, features indicating how good or bad a certain dialog interaction is considered and associated metrics have to be developed. Such criteria and their meaning however may vary from task to task. For example, in a goal-oriented dialog system which provides flight information short dialog interactions would be highly appreciated, while on the other hand a social chat-robot dialog system might consider long interactions as a positive sign for user satisfaction. Once different evaluation criteria have been identified, their influence on the optimization process have to be well balanced, e.g. in the form of weights. In their pioneer work, Levin et al. [12] proposed a generic formula for determining the cost C_d , i.e. negative reward, of a dialog in the following way:

$$(2.18) \quad C_d = \sum_i w_i c_i$$

Hereby, c_i stands for the cost contribution of an identified objective dialog criteria, incorporated to strategy optimization with its associated weight w_i . Typical reward functions of goal-oriented dialogs use the number of turns until task completion and dialog success as common evaluation criteria. However more sophisticated approaches have also been proposed in the past, such as incorporating ASR confidence levels within the optimization function [13]. A further important design issue is if the reward function should be designed as per-turn reward or single reward at the end of each dialog interaction. Depending on the reinforcement learning method used for strategy learning, both approaches have been applied within the past.

2.3.2 Model-based vs. simulation-based strategy learning

There are generally two different approaches facilitating the reinforcement learning of dialog strategies: a model-based approach and a simulation-based approach. The later is, in some literature, also referred to as model-free learning. Underlying ideas as well as advantages and disadvantages of both approaches will be now discussed in more detail.

2.3.2.1 Model-based strategy learning

As already implied by its denotation, the model-based approach necessitates a full model of the environment in order to apply reinforcement learning techniques for strategy learning. More precisely, when following the model-based approach, the state transition function $T_{ss'}^a$, as well as the reward function $R_{ss'}^a$, are estimated from a experienced dialog corpus. Based on the necessary transcription of such experience corpus with regard to the occurred state transitions and generated rewards, the state transition function and reward function are estimated typically using a simple count-based approach. Figure 2.3 illustrates the model-based approach.

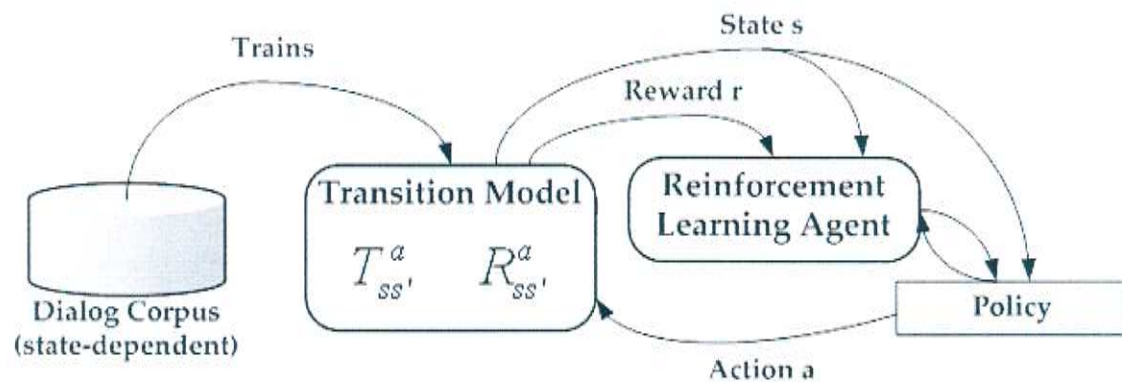


Figure 2.3 Model-based approach for strategy learning

One pitfall of the model-based approach is that depending on the extent of the state space, a relatively big corpus of real dialog interactions is required to train the transition model accurately. The dialog corpus ideally also has to cover all possible action-state transitions several times to escape data sparsity issues. As already mentioned, the dialog corpus has to be transcribed according to the occurred state transitions, which means it is clearly state-dependent. A change to the action-state space configuration would therefore require a costly revision of the dialog corpus and potentially, e.g. in cause of newly added system actions, further require an addition of new dialog interactions to cover the extended state space. Therefore, using a model-based approach profits clearly from an early fixture of the state space configuration. An encouraging aspect of model-based strategy learning is the fact that all dynamics of user and system component behavior are comprised in one single state transition function. Reinforcement learning algorithms can easily be set up and strategy learning can be executed rapidly as dynamic programming techniques are applicable. Examples of work using a model-based approach for strategy learning include [14] and [15], which will be introduced in more detail in chapter 3.1.1.

2.3.2.2 Simulation-based strategy learning

The simulation-based approach differs to the model-based approach in that it does not try to determine directly the state transition function, but rather makes the effort

to model the environment by estimating user and error models for the system components. Interfacing such models with a dialog manager component then allows the simulation of completely artificial dialogs. Figure 2.4 illustrates the simulation-based approach.

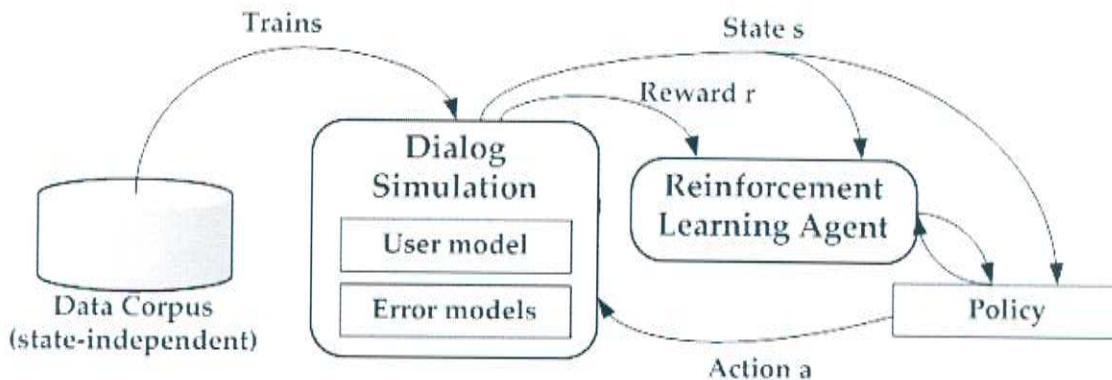


Figure 2.4 Simulation-based approach for strategy learning

The user and error models usually are trained stochastically on a collected dialog corpus, which is transcribed in an adequate manner. Different approaches have been proposed for modeling user and system component behavior which will be discussed in detail at a later point. A key advantage of all simulation-based approaches, however, is that the data corpus used to train the simulation environment is generally independent from the state configuration of the defined MDP. To one point, this allows a greater agility within the learning process as the state space configuration can be changed repeatedly mostly without having any effect or requiring extra work for the transcription of the data corpus. To another point, the necessary extent of the data corpus required to train a robust dialog simulation is typically much smaller than the extent of the corpus needed to train the state transition function as necessary in the case of the model-based approach. Disadvantages of the simulation-based approach include the fact that the set-up and training of several models is often time-expensive and their implementation costly in comparison to the model-based approach. Also, as described before, model-free reinforcement learning of dialog strategies, e.g. through Monte Carlo methods or temporal-difference learning, requires usually more computational effort and is less accurate than the dynamic programming methods available for the model-based approach. Further, explicit modeling of user behavior is a very complex task which can be performed only to a certain extent and with clear restrictions. A moderate accuracy of the simulation environment, however, is a key factor for the optimality of the learned strategy within the real world scenario. We refer to [12] and [16] as exemplary pieces of work using a simulation-based approach. Both research efforts will be introduced in more detail in subchapter 3.1.2 within the context of related work.

Often the amount and characteristic of existing data is what drives the decision to use either a model- or simulation-based approach. In this work, as we promote an

approach to reinforcement learning assuming a small training corpus, we will concentrate on the simulation-based approach.

2.3.3 Dialog Simulation

Having just introduced the simulation-based approach to reinforcement learning, this subchapter will give a short introduction to the field of dialog simulation. While we will delay the introduction of explicit techniques of user and error modeling to the subsequent chapter addressing related work, the focus will now be on presenting the different purposes of dialog simulation as well as the typical setup of a dialog simulation framework. As a final point, we will conclude chapter two by also giving a brief and general introduction to the domain of user modeling.

2.3.3.1 Purposes of dialog simulation

Performing dialog simulation has become more and more popular over the last number of years within dialog systems. The main motivation for dialog simulation hereby lies in the expensiveness and time-consuming character of the collection of real user interactions required within the development, the debugging, and optimization process of dialog systems. Thus, the goal of dialog simulation is to eliminate the need of real world dialog interactions by providing the possibility to artificially generate a theoretically infinite amount of dialog interactions which resemble real world dialog interactions. Hereby the purpose of the generation of such simulated dialog experience can be multifaceted.

One application scenario is the evaluation of dialog systems. Lin et al. [17] develop in their work a handcrafted, rule-based dialog simulation. Through interaction of the dialog simulation with their designed dialog system, they were able to evaluate the quality of their dialog system with respect to several objective dialog metrics. Dialog simulation has also shown to be applicable for dialog system debugging. Levin et al. [12] describe in their work how dialog simulation, and herein especially stochastic user simulation, helped to debug their dialog system in the early development stages by discovering bugs and deadlocks indicated e.g. through unnaturally short or non-ending dialog interactions. A third, and in the context of this work most important, application scenario of dialog simulation is its use for automatic strategy learning purposes. Learning from experience - as propagated by reinforcement learning algorithms - requires numerous dialog interaction samples until the point where a convergence towards the optimal strategy is reached. Usually such extent of experience cannot be collected by online-operation. Instead, a comparable small experience corpus can be used to train a simulation model. The capability of such a dialog simulation model to generate an infinite amount of dialog experience, then allows the successful application of model-free reinforcement learning techniques.

2.3.3.2 Dialog simulation setup

In general, dialog simulation requires individual modeling of each component or actor within the environment that affects the dialog dynamics in any way. So far,

dialog simulation has been mostly applied to mono-modal, speech-only dialogs. For such dialogs, there are mainly two model units that have to be specified: a user behavior model and an automatic speech recognition (ASR) component error model. The generic work flow of a typical dialog simulation setup is illustrated in Figure 2.5. Essentially, the cyclic simulated dialog flow commences with the dialog manager's execution of a particular simulated system action. The user model then responds to that system action by performing the simulation of a single or multiple user actions. In the next step, such user actions are then processed by associated error models which potentially cause a corruption of the original user actions. The eventually altered system actions are then propagated back to the dialog manager component, which, based on this information, executes the next system action.

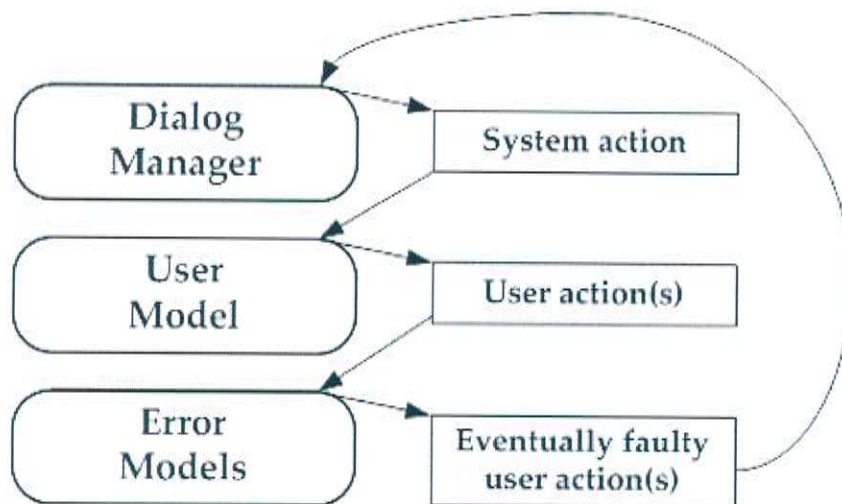


Figure 2.5 Typical dialog simulation work flow

A key issue in setting up a simulation environment is how to design and train the user and error models. In the past, extensive research effort has been expended on proposing and evaluating different approaches, especially in the domain of user modeling. Therefore, in the following subchapter, we give a brief introduction to the generalities of user modeling. Note that a collection of concrete techniques of user, as well as ASR error modeling, will be introduced in detail in chapter 3.2.

2.3.3.3 User modeling

When it comes to simulating a complex behavior like human user dialog behavior, it has to be emphasized that it would be fatuous to assume an appropriate computational implementation of such a multi-faceted process is feasible at all. One must keep in mind that the number of parameters potentially influencing the user's actions within a dialog is simply intractable for building an adequate computational model. The user's current emotional state, his attention level or linguistic capabilities point out only a few of the features which potentially influence his conversational behavior, and thereby would have to be addressed by an authentic user model. Clearly, considering the always restricted amount of training data, such immense

dimensionality is illusive, and therefore any proposed user model can be only seen as very rough approximation of real user behavior.

For the simulation of user behavior there are essentially two options: deterministic user modeling and probabilistic user modeling. Taking a deterministic approach effectively includes developing a hand-crafted and mostly rule-based response behavior of the user, dependant on the current dialog state. In the domain of a typical information retrieval dialog system, such type of user model could be realized, for example, in a way that the user always reacts to system prompts by exactly providing the prompted information, without adding or lacking any portion of information. At first glance, such an approach might seem far too simple, disrespecting the real user dynamics significantly. However, such an approach seems warrantable and practical in scenarios where absolutely no experimental data of task-relevant user behavior exists, but a simulation setup still is desired. Examples of the use of a deterministic user model within dialog simulation can be found in [17], [18] and [19].

While the simplicity of deterministic user models seems inviting, such approach has clear restrictions. These limitations become obvious when we consider, for example, dialog systems using mixed-initiative system actions, instead of exclusively system-initiative actions. Typical dialog opening questions such as “What can I do for you?” usually trigger an immense variation of user responses, which cannot be adequately addressed by a deterministic approach. Instead, for such cases probabilistic and data-driven user models are more sensible. Based on estimated action probabilities trained from collected real dialog interactions, the system simulates conversational user behavior. The expense such necessary prior data collection might have is compensated by a usually far more realistic user model. Hereby one essential necessity is the identification of appropriate parameters based on which user behavior can be trained in a consistent and efficient manner. The choice of a rich parameter set, as e.g. chosen in work by Georgila et al. [20], could include the complete dialog history as well as all information slot states, and thereby provide most likely a realistic and sophisticated user model. However, a drawback of such rich parameter set is that the training of such model requires a huge corpus of dialog data to train the model on. However, in reality usually the available training data for a specific system will be very limited or would have to be collected first. Therefore, in order to challenge data sparsity issues, the parameter set for a user model will normally be chosen carefully and in small extent. Examples for such more simplistic user models are the proposed bi-gram [21] model, the Levin model [12], as well as the Pietquin model [1], which all will be introduced in more detail in chapter 3.2.1.

Depending if a dialog system is designed to perform more than one orthogonal task, a realistic user model would also have to be capable of simulating changes to the user’s intention, as they might occur in reality at any point of time within the dialog. However, data-driven estimation of such intentional changes is challenging as it requires expensive and non-trivial data annotation by hand. Scheffler is presenting a design of such a more sophisticated user modeling in [22]. His approach to user

modeling is also exceptional in way that he uses a combination of deterministic and stochastic user modeling: while goal-dependent actions are performed using deterministic rules, conversational behavior is modeled in a probabilistic way. However, Scheffler's approach to user modeling, as well as the more recent and earlier mentioned approach by Georgila et al. [20], assume the presence of an extensive training corpus, which make them irrelevant within the scope of this thesis.

3 Related Work

The following chapter will provide an overview of previously performed related work in the context of this thesis. Hereby, we will focus on two categories of related work. Initially, subchapter 3.1 will introduce work in the domain of reinforcement learning of dialog strategies. In this context research efforts following the model-based approach as well as the simulation-based approach to strategy learning will be introduced. The relevance and differences of such pieces of work in regard to this presented thesis will be illuminated. Subsequently, subchapter 3.2 will focus on the domain of dialog simulation and therein will chiefly introduce so far proposed techniques for user modeling as well as ASR component error modeling. In such course, the applicability of such techniques within the domain of this diploma thesis will be discussed.

3.1 Reinforcement learning of dialog strategies

In chapter 2.3.2 we introduced the differences between model-based and simulation-based strategy learning. In the following, we will now introduce and describe the so far most important and frequently cited pieces of research work for both approaches.

3.1.1 Model-based strategy learning

In the work of Singh et al. [14] as well as Walker [15], for the first time a research effort of model-based strategy learning within dialog systems is presented. Within both pieces of work, the authors perform automatic strategy learning for the dialog system ELVIS, a system capable of providing information about the current electronic mail spool of a user through a telephone interface. Hereby, the system is able, among other actions, to give an overview of the emails currently located within the user's inbox, by reading only the headers or the complete contents of specifically referenced emails to the user. In both pieces of work, strategy learning is performed based on a corpus of over two-hundred real dialog interactions. Such dialog interactions were collected from earlier online-operation of the system, using

different handcrafted dialog policies. Such policies mainly diverged in the way of used initiative for prompting particular information, such as mixed-initiative (e.g. "What would you like to do?") and system-initiative (e.g. "Would you like to read your newest email?"). Based on the collected dialog corpus and its transcription according to an earlier defined MDP design, the researchers derive an estimation of the MDP's state transition function T , as well as the reward function R . Hereby, within the training process, Singh and Walker prune sparsely visited states completely from the MDP. The reward specification and thereby strategy optimization is performed in the case of Singh only based on the subjective criteria of user satisfaction⁴, while Walker adds the use of objective evaluation criteria as dialog length and task success. Using a standard dynamic programming algorithm (see chapter 2.2.2.3.1), Singh and Walker compute the optimal strategy for the chosen MDP design based on the trained state transition function T and the reward function R . Only in the work of Walker the optimal strategy is subsequently evaluated on a small dialog corpus of 18 dialogs.

In reviewing their work, Singh et al. note that the carried out model-based approach indeed facilitates to automatically learn "sensible" policies, which appear at least by analytical behavior evaluation equivalent to expert-designed handcrafted strategies. The authors also state that through the experimentation with different MDP designs and an analysis of the resulting learned strategies, valuable cues about a proper state space design as well as an adequate reward function design can be obtained. Finally, for the work of Walker, the deployment of the learned strategy within online-operation showed a significant improvement of user satisfaction and task completion in comparison to the used handcrafted strategies.

The work by Singh and Walker suffers generally from the drawback that, in order to perform strategy learning at all, a relatively big corpus of expensive real dialog interactions is required to accurately learn the state transition and reward function. Although Goddeau et al. [23] introduce a method that decreases the required extent of such dialog corpus, challenging data sparsity through the apposition of lower conditioned joint back-off states and actions, the performance of model-based learning is mostly only sensible assuming the existence of an already available extensive online-operation experience. However, also then such approach has its weakness. Optimally, the dialog corpus used to train the state transition function as well as reward function should include all possible state and reward transitions a sufficient number of times, in order to estimate reliable estimates. The dilemma that arises here is that the use of therefore required explorative strategies within online-operation would cause bad user experiences due to the resulting inconsistent system behavior. Using handcrafted rule-based strategies, on the other hand, comprises the risk of triggering only a small subset of the overall possible state and reward transitions. However, performing the later alternative as in case of Singh and Walker, dialog states of optimal expected value could be never visited during training and

⁴ Test subjects were asked after the use of the system if they would use the system again (yes=+1, maybe=0, no=-1).

would be thereby pruned from the MDP. Thus, the guarantee of learning optimal strategy behavior for the given MDP is clearly threatened by the bias of the dialog experience from the used strategy within data collection. A further drawback arises for the use of a model-based approach in a way, that every change to the MDP's state space configuration directly necessitates an adapted new transcription of the training corpus. Considering the required extensive size of the training dialog corpus, such condition likely represents every time a tedious and time-consuming effort.

As the attentive reader might already assume, the model-based approach as performed by Singh and Walker, represents not an adequate choice of strategy learning within the context of this thesis and the therein pursued goals. As a recall, this work strives for rapid strategy learning during the development process of dialog systems, in absence of any initial knowledge or online-operation dialog corpus. Consequently, executing model-based strategy learning with its requirement of an extensive experienced dialog corpus is clearly contra-productive. Further, our chosen simulation-based approach differs from the work by Singh and Walker, in that it comprises an improved agility of the strategy learning process by suggesting the possibility of continuously adapting the state space if required, without the need of re-transcribing the training corpus. Also, the availability of an infinite amount of artificial dialog experience a simulation model offers, facilitates the use of a rich MDP state space, which presumably triggers learning of more complex and stronger dialog strategies than within a smaller state space. In the work completed by Singh and Walker, such rich state space would represent a clear knockout criterion, as it would lead to an explosion of the extent of necessary dialog interactions to accurately train the state transition function.

3.1.2 Simulation-based strategy learning

Within simulation-based strategy learning, two pieces of work are extraordinary with respect to their importance and received citations within the addressed research domain. In the following, we will now introduce such pieces of work.

The work by Levin et al. [12], carried out in 2000, was the first work to practically perform simulation-based strategy learning within dialog systems. Note that two years before it was the same research group which actually pioneered the idea of formulating dialog management as an optimization problem using a Markov decision process [11]. In their work, automatic strategy learning is applied within the ATIS dialog task, an airline database dialog system providing flight schedules, airline fares as well as information to ground transportation. Dialog simulation for such system is facilitated by the design of a probabilistic simulated user following the Levin model, which will be examined further in chapter 3.2.1.2. User action probabilities are partly hand-crafted and partly specified through training, using an already available online-operation dialog corpus of the ATIS system (operated by a handcrafted strategy). Within dialog simulation, Levin et al. do not consider the possibility of recognition errors. Optimization of the dialog strategy is performed in a sophisticated manner with respect to dialog length, cost of information retrieval (the

expected number of tuples retrieved from the database) as well as the data presentation cost (number of records that are presented to the user). Using a Monte Carlo learning algorithm, Levin et al. compute the optimal strategy based on artificial dialog experience. By analyzing and exposing the learned strategy's complex behavior, the authors show that it is possible to formulate a reward function, a state space representation, and a user model in a quick and simple manner in order to learn strategies, similar to or better than handcrafted strategies designed by human experts.

Based upon Levin's work, Scheffler et al. [24] proposed in 2002 a more sophisticated architecture of dialog simulation for the application within reinforcement learning of dialog strategies. As a novel contribution, Scheffler et al. incorporate next to a user model an error model for the speech recognition component into the simulation process. Similar to the work of Levin, the addressed dialog system, providing information about the show times of a cinema, has already been deployed within the real world (using a repeatedly improved handcrafted strategy) at the moment automatic strategy learning is executed. Based on such available extensive dialog experience, Scheffler et al. train their user model and ASR error model. Thereby the user model is designed on an intentional level, using a dedicated user state model to facilitate goal-directed and thereby consistent user behavior. ASR error modeling is performed using a confusion probability distribution, conditioned on the last uttered user intention. In earlier work [22], the authors empirically validate that such simulation setup is capable of generating dialog interactions which resemble the real dialog interactions with respect to selected user goals and dialog length. By interfacing the dialog simulation with the learning agent, strategy learning is performed using Watkins' $Q(\lambda)$ method. As optimization criteria, Scheffler et al. select the length as well the success rate of dialog interactions. An evaluation experiment performed within the simulation environment shows that the learned strategy outdoes the online-operation performance of the used handcrafted strategies. Furthermore, the authors emphasize on the high value of the ease of evaluating different state-space configurations and their effect on the learned strategy performance when performing simulation-based strategy learning.

The work presented by Levin and Scheffler is highly relevant and valuable for this thesis, particularly considering the conformance of using a simulation-based approach to strategy learning. However, for the introduced pieces of work, automatic strategy learning is performed at a point of time where the addressed dialog systems are already deployed in the real world and online-operation is performed using handcrafted strategies. Such available dialog experience is then used for the training of the simulation model. One central goal of performing automatic strategy learning, however, is saving the effort of handcrafting strategies. This goal is obviously not reached within the work proposed by Levin et al. and Scheffler et al.. Our presented work significantly differs from such work, in a way that we do neither require any availability of online-operation data, nor the design of handcrafted strategies at any point within the strategy learning process. As we build our simulation model based on a small data corpus obtained through an initial

Wizard-of-Oz experiment, a discrepancy arises in the size of our training corpus compared to the size of the used dialog corpora in the two pieces of introduced work. Yet, we compensate such a downside by using simplistic and low-conditioned stochastic models for user and error modeling, which we consider sufficiently expressive for the addressed task-oriented and slot-based dialogs as they are typical within human-robot interaction.

A further aspect, in which we diverge from the work by Levin and Scheffler, is that we do not only address monomodal, speech-based dialog interactions, but multimodal dialog interactions for automatic strategy learning. Such effort consequently requires extending the simulation architecture by multimodal capabilities, hereby involving the need of adequate choices for the used user and system error models. Furthermore, a key weakness in the work by Levin and Scheffler is that a proper empirical evaluation of the performance of the learned strategies is missing. While Levin et al. conclude a complex and adequate behavior of the learned strategies only by an analytical strategy evaluation, Scheffler restricts the empirical evaluation of the learned strategy to the simulation environment, and, even more questionable, compares it directly to the online-operation performance of handcrafted strategies. However, an evaluation of learned strategies solely within the simulation environment is always frivolous and does not provide very expressive results. Note that within a simulation-based approach, optimality of the learned strategy is achieved with respect to the designed simulation environment. Practically, however, the simulation model can always only represent an approximation of the real world, while an evaluation of the degree of consistency of the simulation model with the real world is non-trivial. Thus, the performance of a learned strategy within the simulation model is expected to vary from its performance within online-operation, depending on the accuracy of the simulation model. In this context, Schatzmann et al. [25] demonstrate that learned dialog strategies will generally adapt their behavior to any kind of designed user model and thereby, independent of its accuracy or sophistication, will always show promising results when the same user model is also used for evaluation purposes. However, when applying such strategies within the real world, their performances will worsen to an extent depending on the accuracy of the user model. Consequently, proper strategy evaluation as well as comparative strategy studies should always be performed primarily within the real world scenario, in order to guarantee the validity and expressiveness of such evaluations. Within this diploma thesis, we will perform such sort of appropriate empirical strategy evaluation, lacking within the work of Levin and Scheffler, for our concrete exemplary dialog scenario.

3.2 Dialog simulation techniques

In the following, we will now introduce relevant previous research efforts, proposing techniques for user modeling as well as ASR error modeling. In this course, the applicability of the presented approaches within the context of this work will be discussed.

3.2.1 User modeling

In subchapter 2.3.3.3, we already introduced some generalities about user modeling in dialog simulation. In the following, specific proposals for user simulation will be introduced. We will hereby exclusively focus on entirely data-driven and statistical modeling techniques, due to their enhanced suitability within our approach: Entirely or partly deterministic approaches to user modeling, for example as performed within the work by Scheffler et al. [24], oppose our aspiration of a completely data-driven approach to strategy learning, free of any handcrafting procedures. Thus, such sort of proposals to user modeling will be not discussed in the following.

3.2.1.1 Bi-gram model

Eckert et al. [21] first proposed an approach to data-driven user modeling in dialog systems in 1997. In this work, it is argued that the user action is mainly dependent on the complete historic dialog interaction. Correspondingly, it is proposed to condition the probability for a particular user action on all previous user and system actions of a dialog interaction. However, considering the usual availability of only a small data corpus of real user interactions to train such probabilities, it is obvious that such extensive parameter set is not realizable but would lead to immense data sparsity issues. Addressing such restriction, Eckert et al. took the pragmatic but strong assumption that a user action u_t at time t is solely dependent on the immediate preceding system action s_t . Such probability distribution can be then formulated straight-forward as followed:

$$(3.1) \quad p = P(u_t | s_t)$$

In the literature, this approach is also referred to as bi-gram user model. Further, the authors promote a modeling of speech-based user actions on intentional level. For example, given the modality of speech, a particular user action is represented by the user's intention of uttering a specific semantic concept, formulated textually e.g. as *PROVIDE_DEPARTURE_CITY*, instead of simulating a user response on word level (e.g. "I would like to go to Munich"). Generally, the bi-gram model allows the simulation of multiple user actions in response to one system action. A key advantage of the bi-gram model lies in its ease of computation as well as its ability to uniformly model user responses to mixed-initiative as well as system-initiative questions. In the work proposed by Eckert et al., the bi-gram user model is computed (and partly hand-crafted) for a dialog corpus collected within an airline traffic information dialog system called ATIS. At that time, the authors actually did not use the bi-gram user model for strategy learning purposes. Instead, the generation and investigation of a set of 10^4 simulated dialogs facilitated the discovery and the debugging of several significant existent shortcomings within the author's dialog system.

3.2.1.2 Levin model

Depending on the complexity of the dialog system and the number of available user and system actions, using the bi-gram model can easily lead to data sparsity issues. Such condition occurred within the work of Levin et al. [12] and motivated such work's authors to explore a different approach to user modeling, the so called Levin model. For this user model concept more restrictions are put on the user behavior, in order to significantly reduce the extent of the probability table. Thereto, Levin et al. distinguish a user action probability specification for mixed-initiative questions (in their explicit case only represented by the greeting action), system-initiative questions and relaxation prompts. For the case of mixed-initiative questions, three types of probabilities are computed:

- $P(n|s_i)$ - where n is the number of provided attributes in return to system action s_i
- $P(A)$ - represents the overall probability distribution of provided attributes by the user
- $P(V|a)$ - models the distribution of the possible values for a particular attribute a

In case of system-initiative actions, i.e. the system asks explicitly for an attribute, two probabilities are estimated:

- $P(a_u|a_s)$ - represents the probability of providing attribute a_u when asked for a_s
- $P(n|a_s)$ - represents the probability of providing n additional information when asked for attribute a_s (where their distribution is drawn according to the attribute distribution $P(A)$ computed for the mixed-initiative question)

Finally, Levin et al. also model a dedicated probability for potentially existing relaxation prompts as followed:

- $P(yes|a_s) = 1 - P(no|a_s)$ - representing the probability that the user accepts the relaxation for a particular attribute a_s

The Levin model was particularly developed for the use within speech-based dialog simulation in order to facilitate automatic strategy learning. As one can tell by the number of defined probability distributions, the setup and training of the Levin model requires significantly more work compared to the bi-gram model. In fact, it is worth mentioning that in the work presented by Levin, due to the fine-grained probability design, the authors were still forced to handcraft a certain set of probabilities despite extensive online-operation experience. Another pitfall that befalls the Levin model is that it lacks conditioning on the user goal. Prompting for the same attribute twice, might result in receiving two different attribute values from

the user, as the behavior model is not designed to function in a goal-directed manner. Such a drawback comprises the risk of inconsistent and unrealistic dialog interactions.

3.2.1.3 Pietquin model

The goal-inconsistency of the Levin model is precisely where the proposed user modeling technique of Pietquin [1] comes into play. Pietquin proposes an additional conditioning of the probabilities defined in the Levin model on the current user goal. The user goal is defined at the beginning of each dialog in form of a random assignment of a particular value to each existing attribute. In addition to the user goal, Pietquin defines additional environmental probabilities based on what he calls the user knowledge. Among other information, the user knowledge comprises the number of times a particular attribute value has been expressed by the user within the dialog interaction. On the one hand, such user knowledge is used to influence the utterance probability distribution of unsolicited information slots: attributes that have already been provided are less likely to be provided again. On the other hand, the user knowledge is used to model the probability of a user quitting the dialog early, e.g. after having answered the same question repeatedly a certain number of times. Pietquin also introduces priorities of certain attribute values within the defined user goal, determining with which likelihood the user is willing to relax certain given information if asked for. Intuitively, the Pietquin model with its encoded goal-consistency and user state is capable of generating more consistent and realistic user behavior than the Levin model, while at the same time causing a higher probability distribution complexity and ultimately the danger of data sparsity. A significant restriction of the work by Pietquin is that within the work the author did not train or evaluate his user model approach on a real dialog corpus. Instead, user action probabilities were always handcrafted for concrete applications, using common sense.

3.2.1.4 Quantitative evaluation of user models

Addressing the lack of proper evaluation results of the different approaches to user simulation, Schatzmann et al. [26] performed in 2005 a study, comprising a quantitative and comparative evaluation of the bi-gram model, the Levin model and the Pietquin model. Hereby, training of the user models was realized using the huge DARPA Communicator corpus and model evaluation was performed towards two criteria. As a first criterion the researchers investigated how well simulated user responses resembled real user responses for an unseen test set corpus (using a precision and recall metric). The second empirical study analyzed how well the simulated dialog corpora covered the existing variety of user behavior in the real dialog corpus (considering different dialog features such as task length, success rate or dialog efficiency). As a result of their study, Schatzmann et al. show that the Pietquin and Levin model outperform the bi-gram model in a similar extent with respect to both evaluation criteria. Further, Schatzmann et al. also state that, no matter which user model approach is used, still the use of simple statistical metrics suffices to distinguish simulated from real dialog interactions.

3.2.1.5 Applicability of the proposed models in the context of this work

When evaluating the applicability of the introduced user models within the context of this work, it is important to consider the conditions under which the user model has to be set up within our addressed domain and which properties it has to hold. Hereby one crucial criterion is our proposed training of the dialog simulation model on a relatively small dialog corpus. As a direct consequence, our user model has to be designed in a simplistic manner, conditioning the user actions on as few parameters as possible, in order to address efficiently potential data sparsity issues. Clearly, the Pietquin model with its extensive conditioning of the user action probabilities on different user goals and states does not meet such criteria, but instead the use of more simplistic approaches such as the Levin or bi-gram model appear more adequate.

Considering further that we are striving for a rapid setup process of the dialog simulation, another desired property for an adequate user model is its suitability for a straight-forward setup and training process. In this regard, the bi-gram model obviously is advantageous in comparison to the Levin model: The bi-gram model is computed only through a single probability formula, which allows uniform modeling of user actions in response to system- and mixed-initiative system actions. In contrast, multiple probability functions have to be defined and trained within the Levin model to challenge different types of system initiatives and actions. Further, the bi-gram model's simplistic nature facilitates an easy incorporation of multimodality to simulated user actions. While the user action probabilities within the Levin model are designed on an information-based attribute level and thereby particularly tailored for speech-based actions, the bi-gram model offers a greater freedom for modeling multimodal user actions: simulated user actions do not necessarily have to be equated with the provision of a particular attribute but instead, user actions of different modalities and semantic abstraction level can simultaneously be triggered for one system action. Such flexibility is necessary, as within a multimodal dialog interaction, often the individual associated semantic interpretation of a single user action is only determinable within the multimodal context, e.g. after performing multimodal fusion.

Considering the introduced characteristics of the presented user models, it will not surprise that within this work a bi-gram model will be applied for user simulation. Due to the particular circumstances and goals within our work, we deem the bi-gram model to be the best selection for user modeling. Although we acknowledge the introduced comparative results by Schatzmann et al. [26] which make the bi-gram model appear in a rather unfavorable light, it is important to realize that such studies were based on the huge DARPA communicator corpus for user model training. Correspondingly, the publicized supremacy of the Levin and Pietquin model to the bi-gram model is only valid assuming the presence of an extensive training corpus. However, within a typical scenario, and especially within the context of this work, such an amount of training data is by all likelihood not available. Given a significant smaller training corpus, and thereby presumably data sparsity conflicts, we assume

that the shown differences in the performance of the three user models will be scarcely present, if at all. Further, a key criterion for the quality of the bi-gram model is the granularity in which system actions are represented within the user model. A fine-grained specification of user actions, as it will be performed within this work, leads inevitably to a more expressive and qualitative user model. Unfortunately, within the proposed work by Schatzmann et. al, the chosen representation of system actions within the used bi-gram model is not clear. Such condition further impairs the validness of the results of Schatzmann's work for this thesis. A detailed description of our approach to user simulation will be given in chapter 4.3.2.

3.2.2 ASR error modeling

Next to the user behavior, a second component relevant for modeling within dialog simulation is the automatic speech recognition system. While a consensus exists that it is acceptable to assume error-free communication within dialog simulation from the system side towards the user, the opposite direction clearly is error-prone and noisy. However, compared to the attention spent by researchers on user modeling, relatively little work exists regarding the error modeling of the ASR component. While some earlier works within the domain of dialog strategy learning excluded recognition errors completely from dialog simulation (e.g. [12]), other pieces of work spend only little attention to a sophisticated ASR error modeling, for example by simply using a fixed word-based substitution error rate [27]. Within this thesis, two relatively novel and more sophisticated approaches to ASR component modeling will be introduced. The fact that both selected approaches have been proposed by Pietquin et al. ([1], [28], [13]) is an indication of the clearly manageable amount of work carried out so far in this domain.

3.2.2.1 Task-based ASR error modeling

In one approach to ASR error modeling, Pietquin proposes the indication and definition of a finite set of essential recognition tasks within a dialog system. Hereby, recognition tasks are seen as the ASR system's effort of correctly understanding an instance of a particular semantic concept uttered by the user. Depending on the specific dialog system's information space, the format of the addressed semantic information by a recognition task can vary. Defined tasks can involve recognition of plain digits, dates, times, groups of words, continuous speech or other structures. Imagining for example a typical airline information system, hereby one adequate recognition task would be represented by the challenge of understanding correctly the semantic concept of the departure or arrival city. Additionally, further recognition tasks could consist in the recognition of dates or conformational responses by the user. Pietquin suggests that any dialog system can be split into a finite number of recognition tasks, for which then parameters can be specified modeling the ASR system's task-dependant performance. Hereby, in his concrete case, a word-error-rate based approach is used for the task-based error modeling. As Pietquin assumes that for each recognition task the ASR unit's language model is dynamically restricted to the expected semantic concepts, for his approach simulated flawed recognition inevitably results in a confusion of the uttered information within

the same semantic class. In response to earlier work and the use of simplified ASR error models, Pietquin states that a task-dependent evaluation of recognition performance is a far more reasonable and realistic approach to facilitate a more sophisticated ASR error modeling. A shortcoming with the use of recognition tasks is, however, that those tasks tend to be very task-dependent and their definition and performance evaluation has to be carried out for each individual dialog scenario. Addressing such drawback, Pietquin proposes a second, more general, approach to ASR error modeling which will be introduced in the following.

3.2.2.2 ASR error modeling using acoustic similarities

While a detailed description of Pietquin's second proposal to ASR error modeling would go beyond the scope of this work (see [1] for detailed information and formulas), we will focus on communicating its general underlying idea. The approach is based on the contemplation of deriving error-prone recognition performance of the ASR system from acoustic similarities of different word sequences. Such values of acoustic similarities are calculated using an edit distance method based on the phonetic descriptions of the word sequences. A predefined set of nine rules hereby determines the cost of necessary insertion, deletion or substitution actions of phonemes, based on their articulatory features. As an example, one of these nine rules states that insertions or deletions of consonants before or after other consonants are more likely and thereby contribute a smaller similarity distance than insertions or deletions of consonants before or after vowels. Based on such computed acoustic similarity metric, a confusion probability distribution of an uttered word sequence is calculated, specifying the likelihood of its confusion with any other possible word sequence of the underlying language model. Note that within such computation of confusion likelihoods, the probability distributions of the language model as well as the possibility of different phonetic representations of one word sequence are taken into account. Draws of such confusion probability distribution then facilitate error modeling for each possible word sequence.

3.2.2.3 Applicability of the proposed models in the context of this work

Compared to the earlier task-based approach, the immense advantage of using acoustic similarities for ASR error modeling is that no explicit experienced dialog corpus and training process is required in order to compute a confusion probability table. Instead, the error model can be derived directly from the used language model. Its computation within the acoustic-based approach, however, requires the knowledge of optimally all possible phonetic descriptions of every word that exists in the language model, as well as a comparatively expensive implementation effort to compute the acoustic similarities and confusion probabilities. Also, it is questionable how applicable the acoustic-based approach is in the case of not using purely probabilistic methods (as e.g. a standard n-gram language model), but instead context-free grammars within speech recognition (as it is the case in the context of our experimental system setup). Without a doubt, in such cases the design of the context-free grammar with its different parse tree options plays a significant role for

the confusion probability of a certain word segment. While such dynamics could be theoretically be covered by a dedicated probabilistic model, its computation and incorporation within Pietquin's acoustic-based error model would be anything else than straightforward, but require a significant adaption of the approach.

In contrast to the acoustic-based approach, the task-based approach proposed by Pietquin appears independent of the type of used language model. Furthermore, such an approach implicitly covers the mentioned dynamics occurring within the use of a context-free grammar for speech recognition, by learning the recognition performance task-based from direct experience. Also the implementation of the task-based approach is - in comparison to the acoustic-based approach - relatively trivial and thereby promotes our desire of a rapid simulation setup. Especially, considering the use of information-slot based dialogs, the identification of recognition tasks is mostly trivial as usually a recognition task can be defined for each occurring information slot. Finally, the task-based ASR error model allows convenient interfacing with a user model working on intentional level. The acoustic-based user model, in contrast, would require the more complex modeling of user behavior on word level.

The previous arguments motivate our decision to build the ASR error model used within our approach closely related to the task-based approach proposed by Pietquin. However, within this work, we will propose adjustments and enhancements to the original model proposed by Pietquin, in order to improve its fitness for the dialog scenarios relevant in the context of this work. Considering the fact that within our dialog system we do not restrict our language model to particular performed recognition tasks, we extend Pietquin's approach by modeling in-domain as well as out-domain recognition confusion of a semantic concept. Further, as a novel contribution, our ASR model will addresses also potential fluctuations within speech recognition performance between dialog interactions, which have been ignored so far by all previously proposed ASR error models. Chapter 4.3.3 will introduce our ASR error modeling approach in detail.

4 Procedure model

In the following chapter, we will now introduce our developed procedural approach facilitating a rapid simulation-driven reinforcement learning of multimodal dialog strategies in human-robot interaction. Hereby, our approach to strategy learning will be presented in this chapter solely on a conceptual level, while a concrete practical application of our procedure model will be presented in chapter five. As a general remark, it is important that the reader notices that the main contribution of this thesis should not be seen within the definition of the to-be-introduced six different phases of our procedure model and their proposed sequential execution. In fact, most of the defined phases are required for any simulation-driven approach to automatic strategy learning and therefore have been indicated and performed in such way before. Instead, the main contribution of this work lies in the way such different phases are designed, hereby, always spending the focus on the best possible consideration of the environmental conditions and followed goals within the scope of this work.

4.1 Introduction

The process of reinforcement learning of dialog strategies generally involves the sequential execution of several phases. Depending on the chosen approach, some phases might vary significantly (e.g. the training process of either a dialog simulation model or the MDP's state transition function), while others phases will be required and resemble for any approach (e.g. the definition of the MDP). Within this thesis, we attempted to contrive a holistic approach that covers all phases of automatic dialog strategy learning. Hereby such approach was tailored specifically to satisfy the following requirements:

- The approach should address and suit the peculiarities of typical task-oriented and slot-based human-robot dialog interaction.
- Strategy learning should be performable in a rapid and straight-forward manner.

- Learning of multimodal dialog strategies should be feasible.
- The effort of handcrafting dialog strategies at any point should be omitted.
- The approach should be applicable for scenarios where no initial knowledge about the user dynamics or system performance metrics exists.

As a result of this effort, we developed a procedure model for dialog strategy learning consisting of six phases as illustrated in Figure 4.1. Our simulation-based approach covers the entire strategy learning process, commencing with an initial data collection used to train a dialog simulation model until the final process of strategy improvement during online-operation. We will now introduce the six phases of our procedure model in detail while also addressing our motivation for our procedural design decisions within each step.

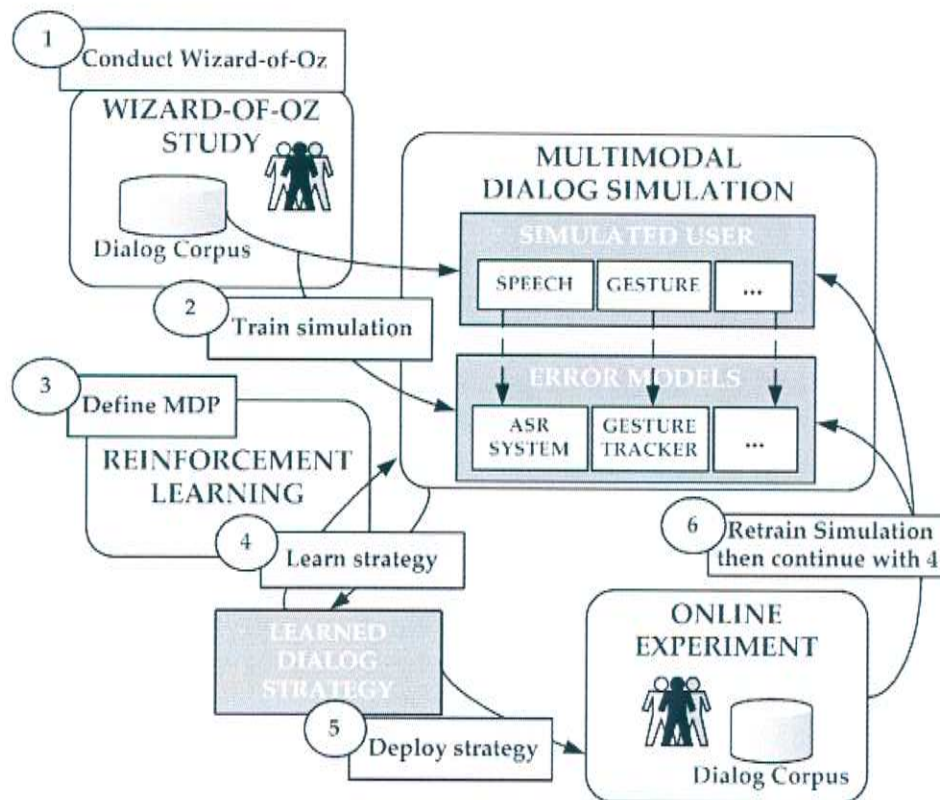


Figure 4.1 Developed procedure model in a step by step manner

4.2 Step 1: Conduct a Wizard-of-Oz experiment

As an initial step of the rapid strategy learning process, we propose the execution of a so called Wizard-of-Oz (WOz) experiment for the addressed dialog system. Generally, a WOz experiment, as introduced by Kelley [29] in 1984, is a valuable instrument within the development process of any kind of interactive computer-aided application. Hereby, a hidden human operator, the so called wizard, simulates partly or completely the behavior of the application under development. At the same time, test subjects are left in the belief of interacting with an autonomous and finalized system. Such an experiment provides, at an early point of development, valuable knowledge about the system-domain, as for example insights into expected

user dynamics or system requirements. In this course, WOz experiments can spare the time-consuming process of developing prototype systems, while providing similar knowledge.

Also within the domain of dialog systems, the execution of Wizard-of-Oz experiments has been popular for early knowledge acquisition. Typical responsibilities undertaken by the human wizard include the performance of the automatic speech recognition task, as well as carrying out the system's dialog strategy during runtime. Ammicht et al. [30] introduce a general framework for such execution of WOz experiments in natural language spoken dialog systems. Further, the use of Wizard-of-Oz (WOz) experiments has been also proposed particularly in the context of reinforcement learning of dialog strategies Williams et al. [31]. In this work, the collection and transcription of a WOz trial dialog corpus is proposed to facilitate a proper action-state space design for the MDP.

As a general guideline, WOz experiments should be always setup with the intention to resemble the targeted final system setup as accurate as possible. Hereby, it is also important that the human wizard simulates his delegated system functionality as realistic as possible. Only then the collected dialog experience within a WOz experiment can be considered representative and qualitative with respect to the final system. Assuming that the human wizard decides directly over the execution of system prompts, it is essential that the test subjects also then are still left in the belief of interacting with a computer-aided system and not a human counterpart. In the work of Jönsson et al. [32] with the expressive title "Talking to a Computer is not like talking to your best friend", the authors note that humans significantly change their conversational behavior when talking to a machine instead of a human counterpart. Addressing this phenomenon, within WOz experiments carried out in the domain of dialog systems, potential direct utterances of the wizard to the users should always be alienated. In the optimal case, such alienation is performed by using the same TTS system intended to be deployed within the final dialog system.

After having introduced some generalities about Wizard-of-Oz experiments, we will now describe the main intentions we follow by our proposed initial application of the WOz experiment technique within the strategy learning process. Recall that within the scope of this work, we assume that no prior knowledge is available about the dialog task domain with respect to user dynamics as well as system performance metrics. While generally it is imaginable that knowledge could be inferred both for the user dynamics as well as the system component performances from previous deployed dialog scenarios, we want to exclude this option in the scope of this work. Instead, we focus on the collection of a Wizard-of-Oz dialog corpus as sole source of task-dependent experience. Hereby, we promote the utilization of the collected WOz dialog corpus in the following ways: training of the dialog situation model, obtainment of valuable cues toward a proper MDP design, and task-specific adaptation and fine-tuning of the language model.

▪ Training of the dialog simulation model

By proposing an initial execution of a WOz experiment, we intend to collect a dialog corpus which is suitable and sufficient to train a dialog simulation model. As introduced earlier, such a training process requires two types of information: firstly, knowledge about the user dynamics within the dialog scenario to build a corresponding user model, and secondly, experience about the performance and error-proneness of the used system components in order to train corresponding error models. We consider the execution of a Wizard-of-Oz experiment as qualified to provide both types of information, assuming an experimental setup capable of tracking therefore all relevant information for their later transcription and evaluation. Within spoken dialog systems, such tracking would primarily include audio recordings of all performed user utterances in order to model speech recognition performance as well as user behavior. Considering the presence of a multimodal dialog scenery, we will need to consider additional recording means. For example, assuming the incorporation of 3D pointing gestures as communication modality, a post-evaluation of the gesture recognizer's error-proneness would require to visually record the test subjects during dialog interaction, using an adequate stereo camera system.

It is important to notice that our approach of considering a relatively small Wizard-of-Oz dialog corpus as sufficient for building a simulation model, is primarily targeted on the particular type of dialog interactions addressed within this work. As already mentioned earlier, typical human-robot interactions exist in the commandment of a single task from the user to the robot and, thereby, usually the extent of the used information space as well as the sphere of possible user and system actions is relatively small. Correspondingly, we argue that given such conditions, often already a small dialog corpus suffices to build a simulation model capable of providing artificial dialog experience from which powerful dialog strategies can be learned. Clearly, a basic assumption of this claim is the use of low-conditioned, but sufficiently expressive, statistical methods for simulation modeling in order to challenge data sparsity. Such methods will be introduced in the second step of our procedure model.

In contrast to earlier work, the use of a Wizard-of-Oz experiment as sole instrument for data collection comprises a key advantage: the design of handcrafted strategies can be fully omitted from the strategy learning process. Recall, that all earlier work within the domain of strategy learning used already available online-operation data for strategy learning. Hereby, such online-operation experience inevitably could only be collected through the use of handcrafted strategies. By using our approach, however, the human wizard within the WOz experiment is able defined the dialog strategy intuitively and spontaneously during run time. Correspondingly, tedious handcrafting of dialog strategies is inhibited and instead, optimized learned strategies can be used from the first moment of online-operation.

- **Obtainment of valuable cues for a proper MDP design**

In conformance with the earlier mentioned work by Williams et al. [31], a second advantage we see in the execution of a Wizard-of-Oz experiment is that such a pre-study is capable of revealing valuable cues towards a proper mapping of the dialog scenario to the MDP formalism. As introduced within the basics part of this work, such mapping is essential to perform reinforcement learning of dialog strategies and requires the definition of an action-state space as well as a numerical reward function. For the design of both components we deem the WOz data corpus to be beneficial.

An advantage of the performance of a Wizard-of-Oz experiment exists in the ability of the human wizard to react to dialog dynamics spontaneously. In such course, it is imaginable that during the execution of the WOz experiment the need for system actions reveals, which might not have been bared in mind by the system developers in the first place. Falling back however on the usual wizard's freedom of system action selection and spontaneous design, it is principally possible within the course of a WOz experiment to dynamically collect experience for unpredicted dialog states and system actions. In a subsequent stage, the system designers can then decide which performed wizard actions to incorporate within the MDP design and which to treat as out-of-domain actions.

However, not only system actions, but many times also valuable cues towards the design of state features can be derived from a WOz dialog corpus. The condition of being confronted with real dialog interactions often facilitates the recognition of environmental features which significantly influence the dialog interaction but might not have been obvious in the theoretical design of the dialog system. Wisely, such observed dialog state features should be incorporated as state features within the MDP design, considering that only through the compliance of the state space design with the Markov property (see chapter 2.2.1) learning of truly optimal strategies becomes achievable.

Next to the action-state space design, we also see the performance of a WOz experiment capable of supporting the setup of an adequate numerical reward function for the MDP. In this context, however, we do not focus solely on the WOz dialog corpus, but propose a coupling of the performance of the WOz experiment with user interviews. Recall, that the reward function should include any metrics towards which strategy optimization is desired. The indication of objective optimization criteria, as the success or the length of a dialog, is straight-forward and does not necessitate data-driven investigation. However, also subjective criteria, such as e.g. the user's perceived naturalness of a particular dialog interaction, principally influence the quality of a dialog and thereby might want to be considered by the system designer within the reward function. However, in order to facilitate such incorporation, a survey of the user's subjective impressions about performed dialog interactions is required. Such knowledge can be obtained from interviewing the test subjects immediately after their interactions with the system. We believe that such collected feedback experience, in addition to objective dialog evaluation criteria, is

valuable for a more sophisticated definition of the reward function. Within the experimental application of our approach, we will give an example of the execution of user interviews coupled to the WOz experiment. We thereby facilitate the definition of a reward function which optimizes a dialog strategy not only to the objective criteria of success and length, but also towards the users' perceived naturalness of the dialog.

▪ **Task-specific adaptation and fine-tuning of the language model**

Last but not least, the initial execution of the WOz experiment can be also of value within the overall development process of the dialog system: the collection of a WOz dialog corpus provides an early insight into the vocabulary and grammatical constructs used by the test subjects within the specific task domain. Thereby a data-driven adaption of the language model of origin towards a more task-specific language model is facilitated. As one example, through such adaptation process the percentage of out-of-vocabulary terms occurring during online-operation can be diminished. Assuming that the user's utterances have been recorded during the WOz experiment, further the opportunity arises that the effect of alterations to the language-model on speech recognition can be directly evaluated. By executing speech recognition offline on the recorded speech samples, the used language-model can be tuned towards desired ASR performance metrics. Within this context however, considering the presumably small extent of the Wizard-of-Oz data corpus, system designers should be cautious not to over-fit the design of the task-specific language model on the WOz dialog corpus.

4.3 Step 2: Train simulation model

In the preceding subchapter, we introduced our concept of training a dialog simulation model solely on the Wizard-of-Oz corpus. We also remarked that in order to address the thereby caused relative small amount of training data, simplistic and low-conditioned statistical methods have to be applied within our simulation model. Within the following, we will now introduce specifically the techniques we propose for dialog simulation within our procedure model. Hereby, design decisions were always made with respect to the following overall requirements of our simulation model:

- The used stochastic methods should be capable of realizing an adequate accuracy of dialog simulation from a small set of training data.
- The setup and training of simulation model components should be performable in a straight-forward manner, promoting our ambition of a rapid strategy learning process.
- The dialog simulation framework has to be capable of handling multimodal user behavior and system error modeling.
- The simulation model components should be suited for the peculiarities of task-oriented and slot-based human-robot dialog interactions.

Based on such requirements we designed a simulation architecture which will be introduced in the upcoming subchapter.

4.3.1 Simulation architecture

Figure 4.2 shows the general architecture of our designed dialog simulation framework. Key components are represented by the user model as well as the error models for the different modalities, which we will shortly describe in more detail. While we use one single user model to cover multimodality of actions on the user side, we propose the use of a dedicated error model for each system processing unit associated to a particular within the dialog interaction present modality. In the following, we will briefly describe the workflow of the simulated user-system interaction within our simulation model.

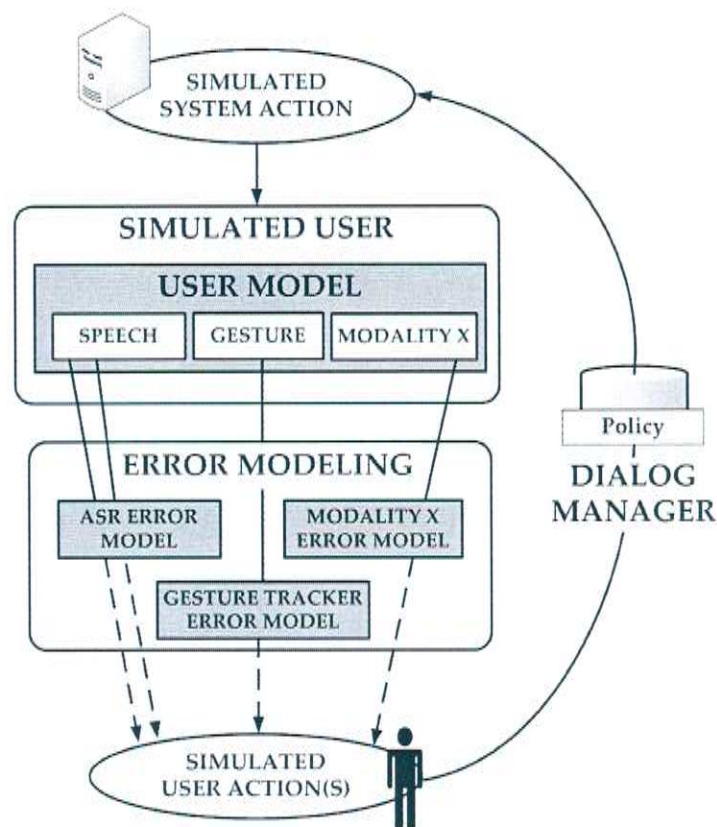


Figure 4.2 General architecture of the developed dialog simulation framework

Triggered by the execution of a simulated system action, the user model probabilistically simulates a set of user actions which generally can consist of zero to multiple user actions. Hereby user action multiplicity can occur within and across the available user modalities. Depending on their modality, the simulated user actions are forwarded to their assigned error models. Basically, three different options exist in which a modality-specific error model can potentially impact a simulated user action. Option one is based on the assumption that no alteration is performed upon the user action, and that the action hereby is handed through the error model in its original version for further processing. In a real scenario, such an option would

represent the perfect functioning of a system recognition component for a concrete case. Considering option two, the simulated user action is falsified by the error model using a model-specific methodology. Clearly, the design and adequateness of such chosen alienation process is highly modality-dependent. Finally, a third option consists in the simulation of a completely unsuccessful detection of the user action by the assigned system recognition component. Given such case, the user action is excluded from any further processing.

Once the user actions triggered by the user model have been processed by their associated error models, the simulated user actions, if detected, are joined and forwarded to the dialog manager component. Such component then carries out the processing and semantic interpretation of the simulated user actions within the maintained dialog discourse. Subsequently, based on the current dialog state and its policy, the dialog manager selects the next simulated system action and the described simulation process starts over. Generally, within the overall design of the simulation model it has to be assured that the used user model, the different error models, as well as the dialog manager can be smoothly interfaced. For example, assuming the modeling of speech-based user actions on intentional level, it is essential that a) the dialog manager is capable of processing and interpreting in such form received user actions and b) the associated ASR error model is working on the same or a compatible abstraction level to simulate error-prone behavior. In this context it should be mentioned that often the dialog manager component used within the simulation domain is individually implemented. Thereby, easy interfacing of the dialog manager with the simulation models as well as efficient computational behavior⁵ can be guaranteed. It turns out that such proceeding often turns out to be less time-consuming and more efficient than trying to customize a dialog management component, usually applied to real dialog interactions, to the needs of dialog simulation.

Having presented the general architecture and process flow of our dialog simulation model, in the following we will now introduce the specific design of our user model and the ASR error model. We hereby will also comment on error modeling of modalities other than speech.

4.3.2 User model

In chapter 2.3.3.3 and 3.2.1 we introduced the concept of user simulation and so far proposed techniques within such domain. Within this course, we also already motivated the use of a bi-gram user model as sensible context of this thesis. Recall that within the bi-gram user model, the user action is solely conditioned on the last

⁵ Considering the eventual necessity of performing millions of simulation runs within strategy learning, the computational expense and duration of the simulation of a single task interaction actually becomes a crucial quality factor for the dialog simulation architecture.

system action (see equation 3.1). We argued that the use of such user model approach is adequate within our procedure model out of the following reasons:

- The simplistic parameterization of the user model facilitates an adequate training process already from a relatively small amount of data.
- The use of a bi-gram user model allows a uniform modeling of different modalities as well as responses to different types of system prompts (mixed-initiative, system-initiative etc.).
- The bi-gram approach offers an ease of model's implementation and training and thereby supports an overall rapid strategy learning process.

In general, the bi-gram user model requires that for each possible pair of system action and user action, a probability value is estimated from a training corpus, expressing the likelihood under which a particular user action is executed in response to an occurred system action. Within the simulation mechanism then samples are drawn from the defined user action probabilities conditioned on the latest system action. Thereby, potentially multiple user actions can be triggered in response to one system action.

As a prerequisite, the system designer must define a certain abstraction level for each present user action modality. For the main modality of speech, we propose, in conformance with the presented work by Eckert et al. [21], the definition of a particular user action on intentional level as the utterance of a semantic concept (SC). Consequently, the number of relevant semantic concepts within a concrete dialog system defines the number of available speech-based user actions. For eventually present additional modalities, other abstraction levels of the user actions have to be found based on the task environment and the application context of such modalities. As an example, assuming the modality of 3D pointing gestures, a particular user action could be defined as the user's execution of a pointing gesture towards a defined location or certain object of interest. Within the modality of emotions, a user (re)action could be represented by the subconscious switch of the user's emotional state, e.g. from neutral to angry.

An important assumption we follow within our approach to user modeling is that the user is always acting in a fully cooperative and goal-directed manner. Hereby the user always consequently follows an initially fixed goal, provides only correct information and is ambitious of fulfilling the task successfully and as quick as possible. Such assumptions usually ease user modeling in a great extent. To one part thereby the concrete content of abstractly designed user actions can be derived from the user's static goal definition. To another part such assumptions inhibit that the user's intentions have to be modeled explicitly within the user model as they are not subject to changes during the dialog interaction.

Although the bi-gram model offers many amenities, it should be noted that also certain information is ignored by the model's simplistic and low-conditioned character. As one example, not covered by the bi-gram model are potential

correlations of the execution probabilities of different user actions in response to a particular system action. Hereby, especially within speech-based user actions usually the probabilities for the utterance of different semantic concepts are highly correlated with each other. Such information however cannot be captured by the bi-gram model.

In chapter 5.3.1 we will present the exemplary setup of a bi-gram user model for our performed human-robot dialog scenario.

4.3.3 ASR error model

Earlier, Chapter 3.2.2 introduced two proposed techniques for the domain of ASR error modeling. Within this chapter we motivated our choice in designing the used ASR error model closely related to the task-based approach proposed by Pietquin [13]. Recall, our rationale for such decision was based on the following aspects:

- The statistical complexity of the task-based ASR error model is low and thus already a small training data is sufficient to build an expressive error model.
- The task-based approach is well applicable for the typical task-oriented and slot-based dialogs within human-robot interaction.
- The implementation of the task-based ASR error model is straight-forward and thereby promotes our intention of a rapid strategy learning process.
- The approach is easily interfaceable with our proposed approach of modeling speech-based user actions on intentional level.

Despite the fact that the introduced task-based approach to error modeling seems to fit our requirements well, we extended the original proposed model by Pietquin by two factors. The first extension is motivated by different environmental conditions of Pietquin's and our approach of performing speech recognition: Pietquin dynamically constrains the speech recognizer's language model fully to responses for the expected upcoming recognition task. Thereby, only user responses including semantic concepts for which the system explicitly asked within its preceding action will be well understood. While such an approach can be helpful in improving speech recognition due to a general smaller complexity of the language model, it also comprises the risk of losing unsolicitedly provided information by the user, not covered by the restrictive language model. Within our domain, we perform speech recognition in a way that we keep the same context-free grammar as a language model for all system prompts. Additional improvement of speech recognition can be achieved by reasonably adapting the weights of grammar rules according to the expected recognition task (as performed by Fügen et al. within [33]). As a result of such different technical setups, simulated misrecognition of a semantic concept within Pietquin's approach results necessarily in a confusion within the same semantic class, while in our approach such a confusion could occur spread over all present semantic classes. Therefore, in addition to the recognition rates of a certain recognition task, we also train probabilities expressing how often our ASR system confuses misunderstood pieces of information inside (In-Domain) or outside (Out-

Domain) the same semantic concept. Note that the hereby three defined probabilities do not necessarily have to accumulate to a value of one. Instead, the potential missing probability fractions does justice to the cases in which the user's utterance of a semantic concept is completely lost. Such misbehavior could be caused, for example, by faulty segmentation or, particularly in our case and the use of context-free grammars, by the return of a non-completed parse tree as best hypothesis from the ASR system.

A further aspect, by which we significantly extend the task-based ASR error model proposed by Pietquin, is the modeling of existent variations of speech recognition performance between different talkers. We claim that such observed deviations in speech recognition have primarily two reasons: a) divergences in language capability and pronunciation clarity of different speakers and b) potentially present fluctuating environmental conditions influencing the speech signal's quality (e.g. variable calibrations of head-mounted microphones). We argue that within the ambition of creating a realistic dialog simulation, such talker-based deviations within speech recognition performance should not be ignored within the ASR error model, but should be learned from the available dialog corpus. In order to model such phenomenon, we measure the standard deviation $\sigma_{RR(RT)}$ of the recognition rate (RR) for each defined recognition task (RT) over all test subjects while further assuming a normal distribution of the performance fluctuations. Correspondingly, within the simulation, we then compute the recognition rate for every simulated dialog and each recognition task individually as followed:

$$(4.1) \quad RR_{\text{Simulation Run}}(RT) = \text{Sample}(N(0,1)) * \sigma_{RR(RT)} + RR_{\text{Overall}}(RT)$$

Hereby the term $\text{Sample}(N(0,1))$ represents a randomly drawn sample from a normal distribution with mean 0 and standard deviation 1, and $RR_{\text{Overall}}(RT)$ the average observed recognition rate for a particular recognition task over all test subjects. By using the same sample value of the normal distribution for the specification of each task-based recognition rate for a particular simulation run, we create a more consistent ASR simulation. In case a high positive value was drawn from the normal distribution, speech recognition works comparably well for all recognition tasks within that simulation run. For a small negative sample value, correspondingly the opposite effect can be observed.

As introduced earlier, we base the actual training of the ASR error model on the audio recordings available from the Wizard-of-Oz experiment. Given the transcriptions of the user's utterances, we perform automatic speech recognition offline on the collected audio samples and evaluate speech recognition performance in the introduced way for each defined recognition task. An exemplary build-up and training of such ASR error model, including real performance figures, will be introduced within the context of our performed experiment in chapter 5.3.2.

4.3.4 Error modeling of modalities other than speech

The design of error models for modalities other than speech is highly modality-specific. Considering imaginable further modalities such as gaze, pointing gestures, facial expressions, emotions etc., the divergences in the way such modalities are recognized and processed are too strong to be handled by a single uniform approach of error modeling. In our performed experiment, we will present an error model for a 3D pointing gesture recognition unit (see chapter 5.3.3). Hereby, our designed error model will be reduced to the use of three parameters trained on a trial experiment: the average error angle of recognized gestures, the standard deviation of such error angle, as well as the recall rate. Within the dialog simulation process, using the recall rate we simulate whether an executed pointing gesture by the user has been detected by the system or not. In the positive case we then falsify the expected pointing gesture by the user according to an error angle drawn from a normal distribution based on the average error angle and its standard deviation. Obviously, while such way of error modeling could be presumably also well applied to the modality of gaze, a completely different approach to the error model would need to be taken for modalities such as emotions or facial expressions.

4.4 Step 3: Definition of the MDP

Subsequent to the training of the dialog simulation model, step three in our procedure model requires the mapping of the addressed dialog scenario to a Markov decision process, in order to be able to apply reinforcement learning techniques. Generalities about the performance of such mapping have been earlier introduced in chapter 2.3.1. As a short recall, the definition of the MDP hereby requires the design of the following two components: the action-state space and the numerical reward function.

In general, both design questions are considered highly task-dependent. In chapter 4.2, we noted that the initial execution of the Wizard-of-Oz experiment often comprises valuable cues about a proper selection of system actions as well as state features. In this context we proposed the performance of user interviews to facilitate the design of an adequate reward function not only towards objective criteria such as dialog success and length, but also towards the consideration of subjective criteria such as the dialog's naturalness.

Referring specifically to the extent and design of state features, we want to introduce two further guidelines that we promote within our procedure model in reference to a promising MDP design. Hereby, the first guideline aims to enhance the quality of the to-be-learned strategy, while the second guideline promotes learning of agile dialog strategies which are capable of adapting to varying conditions of modality processing.

- **Use of a rich state space in order to learn more complex strategies**

Generally, an adequate state space design is a crucial factor for the success of any reinforcement learning algorithm. The exponential growth of the state space with every additional state feature requires a well-thought selection and encoding of such features. Most work so far within the domain of strategy learning propagates a state space design in an as restrictive manner as possible. Hereby, a widely held belief is that a compact and small state space design is a necessary evil within automatic strategy learning.

Yet, we argue that the restriction to an effusive small state space should only then be followed if dialog experience is limited. In such cases, only a small action-state space facilitates reliable estimates of the state values within the MDP. However, for our simulation-based approach, such condition principally does not apply. Instead, dialog simulation allows us the generation of a theoretically infinite amount of artificial dialog experience. Consequently, a rich state space design is not a central concern within our context as long as computational feasibility of the strategy learning process can be assured. Generally, the use of a rich set of adequate state features increases the complexity and sophistication of the learned strategies and also eases the crucial compliance of the defined state space with the Markov property. Correspondingly, within our procedure model, we explicitly encourage the use of an extensive state space, only restricted by reasonable convergence durations of the simulation-based strategy learning process. Chapter 5.4.2 will introduce an example for a generous collection of state features, applied within the strategy learning process for our exemplary performed human-robot dialog experiment.

- **Use of state features to adapt to fluctuating modality recognition conditions**

A second recommendation we give and intends to support a promising state space design, is the use of what we call "recognition-condition features". Within our concrete human-robot dialog interaction, we were able to extract environmental features indicating if or how well different modalities could be recognized at certain point of times within the ongoing dialog interaction. Assuming the existence of such condition parameters within an arbitrary dialog scenario, we propose their encoding within the MDP state space through dedicated state features. Thereby, we facilitate that the optimal learned strategy is capable of adapting its behavior intelligently to the information if or respectively how reliable the recognition of a particular modality is possible within the current state. The thereby addressed fluctuations of recognition conditions may have several reasons. In the following we will introduce two examples of how we utilized the proposed type of recognition-condition features within our to-be-introduced human-robot interaction, in order to exemplify our proposal for a practical application.

Within our later introduced experimental setup, the user explicitly has the option to either communicate with the system by close speech, using a head-mounted close-talk microphone, or distant speech, wherein the speech signal is processed by a remote omni-directional microphone installed on the robot platform. Clearly, the

condition for speech recognition will be improved in the case of close speech compared to a choice of distant speech. By incorporating such condition through a Boolean state feature within the MDP, the strategy is capable of learning adaptive optimal behavior for both scenarios. However, such adaptation is only possible if such conditions are also adequately represented within the simulation framework. Within the example of close or distant speech, such modeling would have to be represented by training different ASR error models for close and distant speech.

Further, we consider our recognition-condition features also helpful to adapt strategy behavior to more subtle environmental dynamics, which might temporarily inhibit or impair the recognition of a certain modality. For example, within our experiment, the recognition of the modality of pointing gestures usually requires that the user's head as well as his hands are continuously tracked by a stereo camera system. It is easy to imagine scenarios where such tracking might be impossible, for example because the user moved temporarily out of the camera scope. Clearly, given such condition, intelligent strategy behavior would inhibit system prompts which particularly request a user action through the temporarily unavailable modality. In order to facilitate learning of such agile strategies, we encode a Boolean condition feature within the state space of our experiment which indicates whether or not pointing gestures are generally trackable within the current dialog state.

4.5 Step 4: Learning the strategy

Once the MDP setup for a particular dialog system has been finalized, strategy learning can be initiated. The general process of simulation-based strategy learning has been introduced earlier in chapter 2.3.2.2. Hereby the simulation model is interfaced with a reinforcement learning agent based on the defined MDP. Dialog simulation and strategy learning is then concurrently performed until convergence towards the optimal strategy is reached.

Design decisions due within the phase of strategy learning are which reinforcement learning algorithm to use, as well as the specification of such algorithm's parameters. In chapter 2.2.2, we introduced the algorithms of Q-learning as well as Watkins' $Q(\lambda)$ method, each technique promoting the amenities of off-policy learning as well as the guarantee of convergence to the optimal strategy for the defined MDP. Correspondingly, it is not surprising that both algorithms have been applied frequently within the domain of reinforcement learning of dialog strategies. While Q-learning requires principally less implementation effort, we endorse the use of Watkins' $Q(\lambda)$ method as learning method. Compared to Q-learning, Watkins' $Q(\lambda)$ method supports earlier strategy convergence especially given an extensive action-state space (see chapter 2.2.2.3.4). In the following, we will introduce our proposed method of implementation and parameter specifications for such an algorithm.

A general advantage of using a simulation-based approach to strategy learning is that exploration moves can be used extensively and without remorse within the strategy learning process. Unlike for dialog interactions with real users, the

simulated user will not be disturbed by inconsistent system behavior caused by performing a highly explorative strategy. On the one side, the sole use of exploration moves allows a fast and entire exploration of all action-state pairs accessible within the defined state space. Conversely, on the other side, frequent exploration moves inhibit the targeted learning process of Watkins' $Q(\lambda)$ method over multiple time steps. In order to credit both parts, we implement Watkins' $Q(\lambda)$ method by starting off learning with a high rate of exploration moves (using a ϵ -greedy strategy with $\epsilon=0.9$) before diminishing such percentage by an exponential decay function in favor of greedy moves over time (with a minimum of $\epsilon=0.1$). Further, in order to explicitly promote early visits of the entire state space, the initialization of all action-state values is optimistically performed with high values. Such value initialization will cause that also greedy moves will first profoundly explore the state space, before the first visits and thereby performed state value updates will indicate the real asset of different states.

Referring to the update function of Watkins' $Q(\lambda)$ method, we set the discount factor γ to one. We hereby address the fact that dialogs are performed within a finite horizon and therefore reward discounting is not necessary. Further, we design the learning factor α to be dependent on the number of times an action-state value has been visited to present. By using again an exponential decay function, α is reduced with every action-state visit (starting from 0.1 and a minimum of 0.001). As a final parameter of the algorithm, λ has to be specified determining how far received rewards should be back-propagated to earlier visited states. Our experience shows that an adequate choice of λ with respect to early convergence and strategy stability is highly dependent on the defined MDP state space. Such phenomenon has also been observed earlier in the work of Scheffler et al. [16]. Acknowledging such condition, we do not propose a specific uniform value for λ , but instead propose an individual state-space dependent comparative study of the performance of different λ values.

4.6 Step 5: Strategy deployment

As a fifth step within our procedure model, the phase of strategy deployment involves the transfer of the learned strategy out of the simulation domain into the real world dialog system. The technical effort and processes required hereby are highly task-dependent and therefore no universally valid guidelines for such process can be given.

As a key prerequisite of such porting process, the dialog manager component of the real world dialog system has to be equipped with the ability to determine the status of all defined MDP state features at any time a system action needs to be triggered. By further granting access to the learned action-state values from the strategy training process (e.g. technically usually realized in a table-lookup manner), the dialog manager component is then able to greedily pick the system action with the highest action-state value for any given state. Such system behavior finally

corresponds to the performance of the optimal strategy within the real world experiment.

4.7 Step 6: Retrain simulation

Last but not least, the final step of our procedure model represents the only optional step within our procedure model but nevertheless represents a valuable feature within our approach. Note that the steps one to five already form a complete approach to strategy learning, based on the data corpus of an initial Wizard-of-Oz corpus. As a key thesis of this work, we believe strategies learned in such way are indeed capable of outperforming non-trivial handcrafted strategies, given the addressed types of task-oriented dialogs within a human-robot interaction. Despite such conviction, situations might occur where either further strategy improvement is desired by the system designer, or strategy adaptation required due to changes to the experimental setup. For both cases, we propose the execution of the sixth step of our procedure model: the retraining of the simulation model based on then available online-operation data. Strategy improvement, or respectively adaptation, is subsequently available by reinitiating the strategy learning process based on the refined simulation model. Hereby the significance of strategy improvement or adaptation will be determined by the extent the simulation model was altered through the retraining process. Generally, system designers can obtain a valuable cue of the effect of simulation retraining by comparing the state-dependent behavior of the strategies learned before and after the simulation. A good comparison criterion is to observe the percentage of dissenting optimal actions of all states for both strategies.

A decision the system designer has to make within the process of retraining the simulation is which data corpora to include within the training process. Assuming a static online-operation setup, as well as a Wizard-of-Oz experiment that matched adequately the final experiment setup, online-operation and WOz experience can be merged for simulation retraining to obtain a larger and thereby more expressive corpus. Occasionally, however, the environmental conditions of a dialog system change over time. Typical triggers for such environmental changes are modifications to system components. For example, the automatic speech recognition component could undergo a version update or even be replaced by a completely different system. As a result of such process the performance of the addressed system component presumably changes, and correspondingly the so far trained simulation model loses partly its validity. Precisely for such conditions, the performance of simulation retraining – now based on online-experience collected under the new environmental conditions – is highly advisable to further on ensure the optimality of dialog management. Note that hereby not necessarily all previously collected information should be disregarded within the simulation training processes. For the given example the performed change to the experimental setup would only affect the ASR system's performance. Correspondingly, the user model could still be trained on all available dialog experience, while the ASR error model should be trained solely on the latest online-operation experience.

A different methodology that is often proposed for adapting strategy behavior to environmental changes during online-operation is the technique of so called online learning⁶. In the following, we want to briefly present the arguments why we consider the introduced execution of simulation retraining superior to the technique of online learning within our domain:

- In order to allow online learning to have a noticeable effect during online-operation, a relatively small state space has to be chosen. As already noted earlier, we believe that thereby a weaker strategy will be learned compared to the use of a larger state space as possible in the simulation environment.
- Online learning requires immediate knowledge about the fact if a dialog interaction ended successfully or not. Unlike within the simulation domain, it is non-trivial to derive this information in an unsupervised manner directly online.
- In contrast to our simulation-based approach where the strategy is deployed in a greedy manner, online learning requires a continuous use of exploration during online-operation which has a negative effect on online performance as well as user amenity.
- Online learning requires a fixed state space during online-operation. In contrast, using our approach principally allows an alteration of the state space with every new simulation retraining process.

⁶ Within online learning reinforcement learning is performed directly during online-operation, which means action-state values are updated after every performed dialog which includes that the optimal strategy can change from one dialog to the other.

5 Experiment

In the following chapter, we will present the application of the developed procedure model for a concrete human-robot dialog experiment. In this course we commence with a descriptive introduction to the task environment, the used system components, as well as the dialog setup. Subsequently, we will then clarify how the concepts defined within our procedure model found their representation within the context of the strategy learning process for our performed dialog experiment. Note that while intermediary results, as e.g. the probability tables of the trained dialog simulation models, will be presented within this chapter, any presentation and discussion of results referring to strategy performances – both for the simulation and the real world scenario - will be discussed in chapter six.

5.1 The task

5.1.1 Introduction and background

The design of our human-robot dialog experiment originated out of the context of the collaborative research center SFB588 at University of Karlsruhe, developing cooperative and multimodal humanoid robots. The center was established in the middle of 2001 by the “Deutsche Forschungsgemeinschaft” (DFG) with the long-term goal being to assist the human’s every day life through humanoid robots. Hereby the focus of application so far has been the kitchen environment. Within the project, several different robot platforms are engaged in ongoing research activities. The figurehead robot of the research center is ARMAR [34], shown in Figure 5.1. ARMAR’s action sphere is defined by twenty-five mechanical degrees-of-freedom. Roughly introduced, it consists of an autonomous mobile wheel-driven platform as well as two anthropomorphic arms with attached gripper devices. Furthermore, the robot is equipped with an agile stereo camera head and capabilities for multimodal language processing and speech synthesis. While our specific task scenario is targeted for a future online-operation using ARMAR, the performed experiments

within the project of this thesis were carried out on a backup development platform of ARMAR. The concrete composition of such platform will be introduced in the following subchapter.



Figure 5.1 SFB588 robot ARMAR



Figure 5.2 Photograph of the experimental setup

5.1.2 Task description

In our particular dialog scenario, our robot performs the role of an early-stage bartender. The potential bar counter is represented by a table, behind which our robot system is statically placed. In order to simulate different order options for a potential customer, placed on top of the table are twenty objects, diverging in color and shape to facilitate their automatic visual recognition. The types of objects used are ordinary items from the kitchen domain and are represented by plates, cups and bottles. Hereby, attention is spent to the fact that the colors of the objects are clearly discriminatory for the users and the robot in a way that the colors of red, blue and yellow were used. A photograph of the final experiment is shown in Figure 5.2.

The task interaction for our experiment is based on the assumption that the interacting test subject desires a particular item from the table setting, which the robot is supposed to serve to him⁷. Correspondingly, at the beginning of each interaction, test subjects are told to silently choose a specific object from the table setup. Subsequently the robot then initiates a multimodal dialog interaction with the user, attempting to identify the user's desired object from the table.

In compliance with the robot system ARMAR, our backup robot platform is also equipped with a stereo camera head movable by a pan-tilt unit. Also both robot platforms are of similar height. Differences between the two systems however exist with respect to the fact that our robot system is not suited and targeted for changing its location during user interaction. Also our backup system is lacking the two-arm

⁷ Note that due to the lack of an arm system on our backup robot platform, the actual process of serving the item to the user was impossible and scheduled for a later development stage.

system of ARMAR, which is why we installed a laser pointer on top of the pan-tilt unit, in order to still be able to reference particular objects of the table in a visual manner. The dialog capabilities of our used robot platform are characterized by its use of our speech recognition toolkit Janus [35] with an IBIS decoder [36], the dialog management framework TAPAS [37], as well as a 3D pointing gesture recognizer [38]. The TAPAS system and the gesture recognition component will be now described in more detail, before we examine the designed dialog structure of our experiment.

5.1.2.1 The dialog manager TAPAS

Within our dialog system, we use the TAPAS system for dialog management purposes. TAPAS is a language- and domain-invariant dialog management tool developed at University of Karlsruhe and can be seen as an implementation of the dialog algorithms developed within the ARIADNE project by Denecke [39]. The TAPAS system particularly features a rapid dialog system setup. Among other aspects, the setup procedure is eased by the facile access of relevant external system components to dialog management, as e.g. as for example database binding or speech synthesis, through the provided application interface of TAPAS. Within the scope of this work, the key strength of the TAPAS framework has to be seen in its ability to handle multimodal dialog scenarios. For the modality of speech, processing is performed based on the use of context-free grammars. Further, the TAPAS architecture is particularly capable of processing and interpreting 3D pointing gestures. The formalism of typed feature structures (TFS) is used to represent and unify received semantic inputs of different modalities within discourse management. More information about the TAPAS architecture can be found in work by Holzapfel [37].

5.1.2.2 3D pointing gesture recognizer

The pointing gesture recognizer used within our system was developed and provided by Nickel et al. [38]. For such component, gesture recognition is based on a stereo camera signal and on the continuous determination of the 3D-positions of the user's head and hands. In order to facilitate such localization, skin clusters are identified within the visual signal by using trained models of skin-color distributions and morphological operations. By considering spatial depth and height information provided by the stereo video signal, hypotheses can be formed for the most likely assignment of the user's head and hands to such 3D skin-pixel clusters. Hereby also historical information about identified head and hand positions in the preceding time steps is considered. Figure 5.3 shows the application's manner of visualizing a sample video frame within a 3D world, given the correct recognition of the user's head and hands.

The essential gesture recognition process is performed on the spatial trajectories of the hands. Hereby the system decomposes the execution of a gesture in three different phases: a begin phase, a hold phase, and an end phase. Each of the phases is modeled by a Hidden Markov Model using the position of the tracked hand as an

input feature. In order to design such HMMs invariant to the user's absolute position, hand coordinates are transformed to a cylindrical, head-centered coordinate system. Once a hand movement has been detected as a valid pointing gesture, the gesture information is comprised by the specification of two vectors: a) a position vector describing the spatial location of the hand in the moment the gesture was detected and b) a direction vector, comprising the spatial orientation of the detected gesture. Hereby such direction vector is computed by extracting the line from the center of the head to the center of the pointing hand within the hold phase.

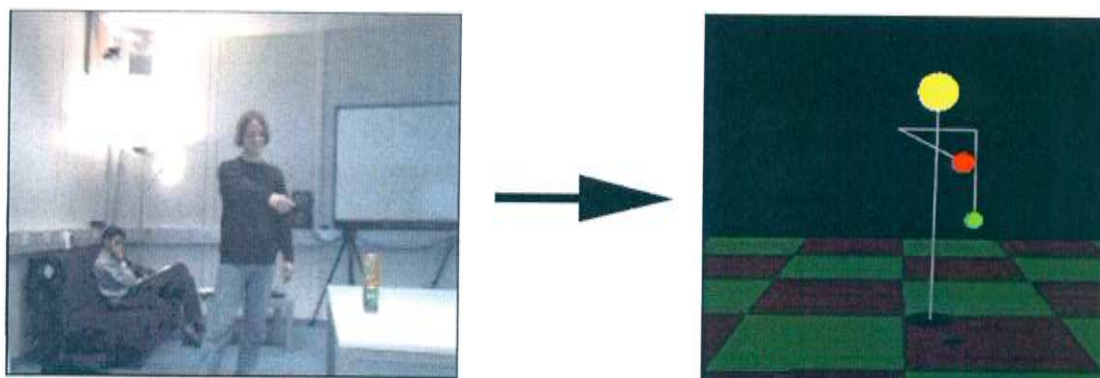


Figure 5.3 Visualization of a video frame within the 3D world of the gesture recognizer component [40]

5.1.3 The dialog setup

The setup of the dialog process will now be described in more detail. Figure 5.4 demonstrates the dialog automation of our performed experiment. As the illustration shows, the robot always initiates the dialog with a self-introduction and the mixed-initiative action "What can I serve you?". After a user input has been received, the robot has the option to prompt directly for one of the three available object attributes represented by the object's type, the object's color and the object's location. For the case of the location attribute, our robot prompts for a pointing gesture of the user towards the desired item. Further options of the robot exist in confirming the speech-based information slots of object color and object type individually or jointly. For the information slot of object location, no explicit confirmation move is available. Note that the introduced actions underlie certain preconditions, as presented in Table 5.1.

During the dialog interaction, the robot maintains a list of candidates including all objects which match the so far the provided information of the user about his desired item. Self-evidently, at the beginning of each dialog all objects are included within the candidate list. As soon as information has been received about the object's color or type attribute, the candidate list is correspondingly restricted. Considering the recognition of a gesture input, we allow a certain error distance of the gesture pointing vector to the object's real location to still consider these objects as valid candidates. However, the list of valid candidates is sorted according to the candidates' proximity to a potentially received pointing gesture.

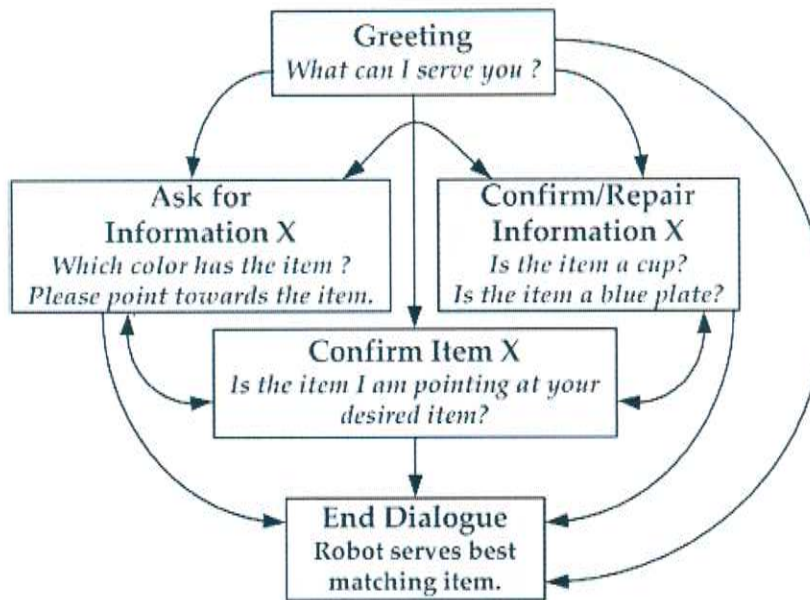


Figure 5.4 Dialog system automaton

As a further system action, at any point of system initiative within the dialog interaction, the robot is able to point towards the so far best ranked candidate on the table, and prompt the user if the referenced item is the one he wishes for. If the user confirms, the robot ends the dialog with the next move and declares the confirmed object as the one to be served. In the case where the question is answered with a negative response, the specific object is excluded from the candidate list. Further, the robot also has the option to end the dialog at any time once the greeting action has been performed. In the moment the dialog is ended by the system, the top-ranked candidate is declared as the item to be served. In case of ties, e.g. only the object color has been acquired and several objects of that color exist, the chosen item is randomly selected among the candidates. Note that also in a state where the candidate pool is empty, the actions of confirming an explicit item or ending the dialog are available. In such cases, the reference or to-be-served object is randomly selected among all existing objects on the table.

Within the final system setup, each system action was combined with a time window of six seconds for the user to answer. If no user response was received in that time the next system action was triggered. Also, we restricted the number of total system actions to a maximum of ten. Once that number was reached, the robot automatically ended the dialog independent of the current dialog state.

Concluding the description of the task setup, Table 5.1 provides a comprehensive overview of all system actions available to the robot including their defined associated abbreviation term, a short description, as well as the preconditions for the action's applicability.

Action	Description	Precondition
<i>Greeting</i>	Self-introduction and mixed-initiative action "What can I serve you?"	Only available at the start of the dialog
<i>AskType</i>	Ask for the object type	Slot object type is empty
<i>AskColor</i>	Ask for the object color	Slot object color is empty
<i>AskGesture</i>	Ask for a pointing gesture towards the desired item	Slot object location is empty
<i>ConfType</i>	Confirm information of object type	Slot object type is filled but not confirmed
<i>ConfColor</i>	Confirm information of object color	Slot object color is filled but not confirmed
<i>ConfAllNI</i>	Confirm object color and type jointly	Slots of object type and object color are filled but not confirmed
<i>ConfBest</i>	Confirm and point at best matching candidate so far	None
<i>End</i>	End dialog and serve best matching item	None

Table 5.1 Available system actions and their preconditions

5.2 Wizard-of-Oz experiment

In the following we will describe the performed execution of the Wizard-of-Oz experiment for the introduced dialog task. In chapter 4.2 we elucidated the purpose of the execution of the Wizard-of-Oz experiment as initial step within our defined strategy learning process. In this chapter will concentrate on the introduction of the experiment's technical setup. We will further present the results of the performed user interviews coupled with the Wizard-of-Oz experiments, as well as discuss why a second data collection study had to be performed.

5.2.1 Experimental setup

As a central goal, the setup of our Wizard-of-Oz experiment was intended to be designed as close as possible to the anticipated setup of the final experiment. However, the experiment's design was significantly influenced by the fact that our robot platform, including the stereo camera system, was not available for the WOz experiment out of the following reason: the first part of this diploma thesis and thereby the WOz experiment was performed at Carnegie Mellon University, USA. However, the robot platform used within the final experimental setup is maintained at University of Karlsruhe, Germany. We compromised for such condition by including next to the human wizard, a second human operator within the experimental setup who was responsible for the simulation of the humanoid robot. In order to give the users at least to some extent the feeling of communicating with an autonomous robot system, the human robot did not interact directly with the test subjects, but instead only reacted upon commands received from the human wizard. The output of any system prompt was performed using a TTS system and forwarded

directly to the test subject's headphones. Communication between the wizard and the human robot was realized through a video-display goggle system that the human robot carried. The integration of such system component allowed the human wizard to send commands to the robot in an undetectable manner for the test subjects. Delegated actions to the human robot were isolated to the tasks of pointing towards a certain object at a given point of time and respectively serving a certain object from the table to the user at the end of each dialog.



Figure 5.5 Photograph of the Wizard-of-Oz experiment (human robot (left), test user (right))

During the experiment, the human wizard had the task of simulating the system's functionality of speech recognition, 3D pointing gesture recognition, as well as system action selection. As no technical capabilities were present to transfer a real-time video and audio signal of the experiment setup to a shielded wizard location, it was necessary to place the human wizard within sight and hearing distance of the test subjects. We choose the wizard's location to be a laterally shifted behind the test subjects, in order to stay outside of the test subject's direct view. In order to facilitate the performance of the wizard's tasks, a special GUI application was designed. Such application allowed the wizard to capture observed speech as well as gesture user actions in a textual manner. The GUI application was interfaced with the TAPAS system for discourse management and TTS linkage. In order to facilitate fast system responses through the wizard, the execution of predefined system actions (as shown in Table 5.1) was provided through a button interface. Yet, besides the availability of system action selection through the GUI interface, the wizard was principally also able to spontaneously output any arbitrary system prompt to the user by using a dedicated text field within GUI application. An option, which, however, was nearly never used by the wizard as it turned out during the experiment's execution that the predefined system actions sufficiently covered the occurred dialog dynamics. With respect to the chosen dialog strategy specified during runtime, the wizard followed a policy which intended to collect audio and interaction samples of the users for a wide range of different dialog states. Among others, such behavior involved the frequent use of conformational questions as well as the deliberate incorporation of recognition errors through the wizard.

In order to record all relevant information for the later transcription and training of the simulation model, we taped all dialog interactions with a customary digital camera system as well as recorded all user utterances on audio level. The video material was intended to ease the transcription of occurred speech and gesture user actions within the context of the preceding system action, which represented the essential information for the training of the user model. Further, audio recordings were performed in two manners. To one part, users were equipped with a head-mounted microphone for the recording of close talk samples. To the other part, we installed an omni-directional microphone below the table top to record distant speech samples of the users. Automatic speech recognition was then performed offline on the recorded samples for ASR error modeling. The audio recording process hereby was performed by the laptop which was also used for the operation of the wizard's GUI application. Figure 5.6 shows a sketch of the WOz experimental setup.

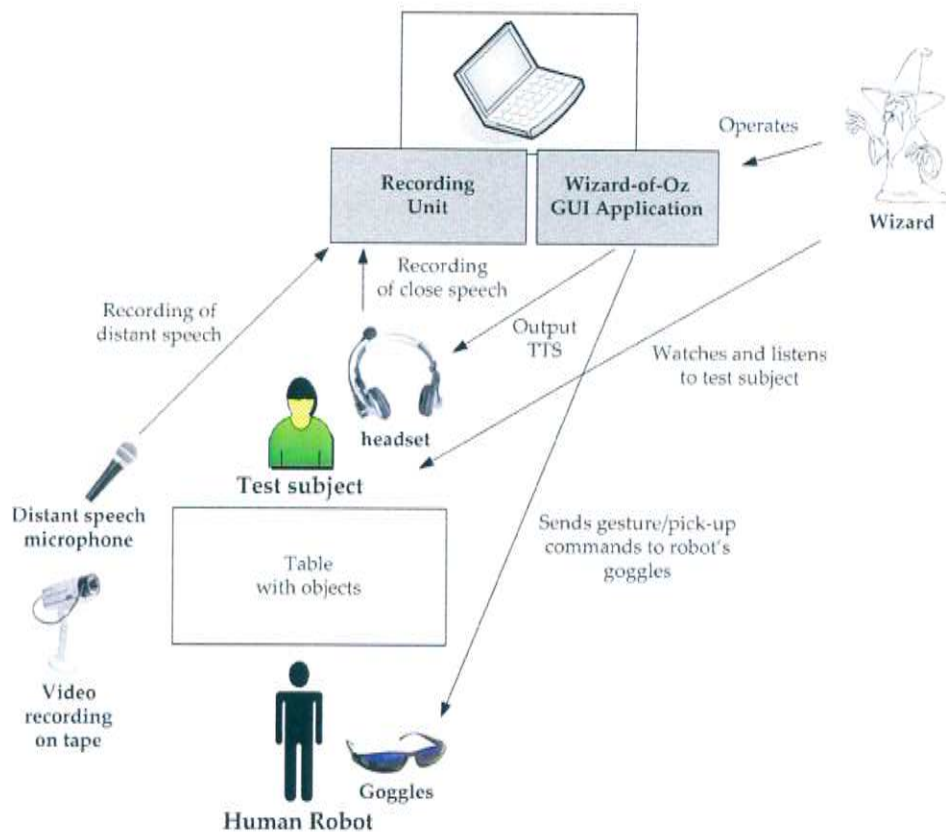


Figure 5.6 Sketch of the experimental setup of the Wizard-of-Oz experiment

5.2.2 Experiment execution

Within the execution of our Wizard-of-Oz experiment 15 test subjects were engaged, including both native and non-native English speakers. In order to build up a common ground of knowledge of each test subject, the users were asked to read through a prepared information sheet including a scenario description before the first dialog performance. A copy of this sheet can be found in Appendix A. On average, each test subject performed five to six dialogs. In total, 82 dialogs and 314

user utterances were collected. Listing B.1 within the Appendix shows a sample dialog of the Wizard-of-Oz experiment and provides an insight to the required operations of the wizard and the human robot within a typical dialog interaction.

5.2.3 User interviews

For every test subject, user inquiries⁸ were performed during and after the completion of the Wizard-of-Oz experiment. To one part, after every dialog interaction an interrogator prompted the test subjects for their subjective impression about two parameters of the dialog: firstly, how natural the preceding dialog was perceived, and secondly, how adequate the length of the dialog was considered given the particular task. For both features, a value on a scale between -2 and +2 was expected.

By asking for the perceived naturalness of a particular dialog, our intentions were to receive indications of unnatural system behavior. Thereto, we originally planned to investigate eventually existing correlations of the execution of certain system actions or combinations of system actions resulting in a poor evaluation of the perceived dialog's naturalness by the users. However, our experience showed that our ambitions in this context were set too high. It appeared that users were not able to indicate the naturalness of each individual dialog in such isolated manner. Instead of evaluating the preceding particularly performed dialog interaction, users tended to evaluate once the perceived naturalness of the overall experimental setup, and then returned such impression for every performed dialog interaction. One indication for such condition was among others things the fact that for all test subjects the returned values for naturalness of one user were equal or diverged only slightly over all performed dialog interactions.

However, fortunately we still were able to receive cues about perceived unnatural system behavior through another way during the user survey: During free conversation at the concluding "general comments" part of the interview, several test subjects considered particularly two types of behavior patterns of the system as unnatural: firstly, the direct repetition of the exactly same system action within the dialog flow, and secondly, a repeated prompt for a particular object attribute, performed by the system e.g. due to speech recognition errors. Although, both behavior patterns might be attractive in particular situations from a system point of view, their artificial character lead to an overall reduced user satisfaction and thereby diminished quality of the dialog interaction. Consequently, the incorporation of such knowledge within the strategy optimization process and thereby the MDP's reward function, as shown within chapter 5.4.1, is only sensible.

Compared to the performed direct prompts for the naturalness of concrete dialogs interactions, our decision to prompt for the adequacy of the dialog length comprised higher information content. Figure 5.7 shows the results of such user interrogation.

⁸ A copy of the underlying survey form can be found in Appendix A.

The evaluation shows that dialogs which required four to six system turns were still considered as of adequate length by the users, indicated by the associated positive average scale value. However, for dialogs requiring seven and more system actions, the obtained average scale values secede below zero, implying that the users would have deemed a shorter dialog interaction as adequate for the given task. Such collected user feedback was useful when specifying the weight of the criteria of dialog length within the MDP's reward function. Self-evidently we designed the weight in a way that the resulting dialog interactions using the learned strategy in average did not require more than six systems actions.

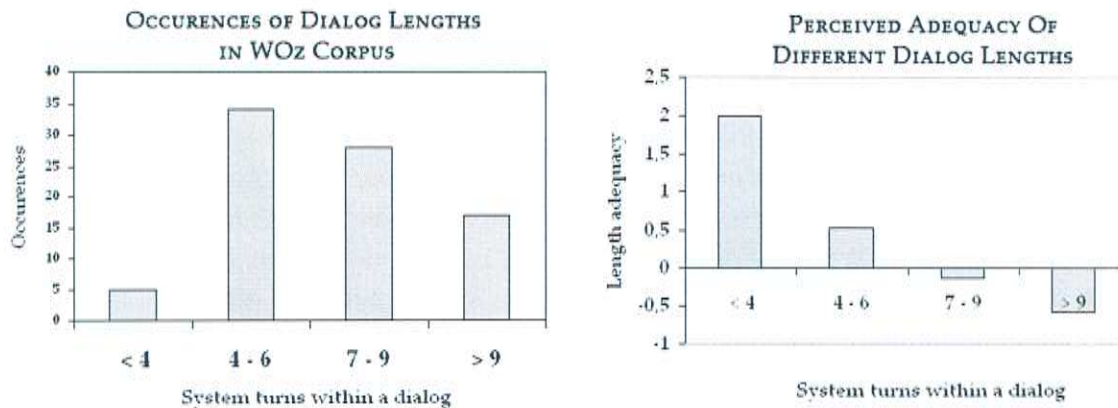


Figure 5.7 Evaluation graphs of the performed user interviews

As a further part of our user study, we asked our test subjects two more questions intending to collect some general feedback of the users about the incorporation of gestures within the given dialog task. Hereby, within the first question we wanted to evaluate if the use of pointing gestures within the given setup is at all considered as appropriate by the users. Thereto the users were asked how beneficial they considered the use of gestures within the given task domain in general. We asked for a value from a scale of -2 to 2 where -2 represented that the incorporation of gestures is useless while +2 indicated that such incorporation is very beneficial within the given dialog scenario. Indicating a clear positive response to the adequateness of the incorporation of gestures, the average returned value to this question was +1.53 with a standard variation of 0.63.

We further wanted to know from the test subjects how natural they felt about being explicitly asked from the system for a pointing gesture towards the desired object, compared to being asked for providing the object attributes through the modality of speech (e.g. *AskColor*, *AskType*). Our motivation for such question was that we feared that direct prompts for pointing gestures would be perceived as unnatural or artificial from the users compared to speech-based system actions. Again a value on the scale between -2 and 2 was expected, while -2 represented prompts for gestures being more artificial than prompts for a speech input (vice versa for the value of +2). The average observed value was against our initial assumption slightly favorable to the system action of *AskGesture* with +0.87 and a standard deviation of 0.99. As

response to such observed feedback, we dismissed our initial plan of explicitly penalizing the execution of the system action *AskGesture* within the MDP's reward function. We initially falsely considered such action as beneficial to optimize our dialog strategy towards a higher naturalness.

5.2.4 Additional data collection

Unfortunately, the performed Wizard-of-Oz experiment at Carnegie Mellon University did not suffice for the collection of all necessary information for dialog simulation, as actually propagated within our procedure model. The reason for that was of logistical nature: Optimally, all test users should have been recorded during all WOz dialog interactions by a stereo camera system compatible with the introduced gesture recognizer component. An offline processing of such video material then would have allowed the training of the required error model of the gesture recognizer component. However, given that such a stereo camera system was not available on-site at Carnegie Mellon University during the WOz experiment, a second small data collection experiment was setup at University of Karlsruhe. In order to collect suitable and task-relevant data for evaluating the gesture recognizer performance, the conditions of the final experimental setup were reproduced as accurate as possible. Thereto, we placed our – at this point of time available - robot platform including the stereo camera system statically behind a table in the same way as performed within the final setup. However, this time only three objects were placed on the table, evenly distributed over the table's width. During the experiment's execution, the test subjects received continuously prompts over a screen, requesting the performance of a pointing gesture towards particularly one of the three objects on the table. The gesture recognition was performed directly online and the collected data corpus then used to train an error model for the gesture recognizer in the way it will be introduced in chapter 5.3.3. In total, seven test subjects participated in the data collection and a total of 93 pointing gesture samples were collected.

5.3 Training of the simulation

Having finalized the process of data collection as introduced in the preceding subchapter, all requirements were met for the training of the simulation model. In chapter 4.3 we introduced our simulation architecture as well as the concrete probabilistic methods used for the modeling of the user behavior as well as ASR performance. Now in the following, we will introduce the concrete computed probability distributions for such methods trained on the collected WOz dialog corpus. Additionally, we will introduce our approach to setting up an error model for the gesture recognizer component as well as the results of its training process.

5.3.1 User model

As introduced earlier, our procedure model proposes a bi-gram approach to user modeling where the likelihood of the execution of a particular user action is solely conditioned on the preceding system action. Therefore, the dimensions of our probability table is represented by the number of available system actions times the number of possible user actions. As an exception, the system's option to end the dialog is not represented within the probability distribution. As such system action does not include any further processing of user actions it can be excluded from the user model. From the user side we indicated four different types of user actions within our task environment. Three user actions refer to the modality of speech, representing the user's utterance of the semantic concept of object type, object color or conformational responses. As a fourth user action, the user's execution of a pointing gesture towards his desired item was indicated. Table 5.2 shows the computed probability distribution based on the Wizard-of-Oz corpus. Recall that descriptions of the abbreviated user actions can be found in Table 5.1.

Action	Type		Color		Gesture		Confirm	
<i>Greeting</i>	67/82	81.7%	51/82	62.2%	21/82	25.6%	0/82	0.0%
<i>AskType</i>	30/35	85.7%	2/35	5.7%	1/35	2.9%	0/35	0.0%
<i>AskColor</i>	1/42	2.4%	38/42	90.5%	0/42	0.0%	0/42	0.0%
<i>AskGesture</i>	0/60	0.0%	0/60	0.0%	59/60	98.3%	0/60	0.0%
<i>ConfType</i>	0/65	0.0%	0/65	0.0%	0/65	0.0%	65/65	100.0%
<i>ConfColor</i>	0/62	0.0%	1/62	1.6%	0/62	0.0%	62/62	100.0%
<i>ConfAllNI</i>	0/21	0.0%	0/21	0.0%	0/21	0.0%	21/21	100.0%
<i>ConfBest</i>	1/102	1.0%	0/102	0.0%	0/102	0.0%	102/102	100.0%

Table 5.2 Bi-gram user model learned from Wizard-of-Oz dialog corpus

How to read the table:

The table states, for example, that as a response to the mixed-initiative system *Greeting* in 62.2% of the times the information about the desired item's color was provided by the user. Further, assuming that the system asked for the object type, in 2.9% of the times users pointed towards the object while in 85.7% of such cases they returned the solicited type information.

5.3.2 ASR error model

Following our approach of a task-based ASR error modeling as introduced in chapter 4.3.3, every essential semantic concept present within the dialog setup was mapped to an individual recognition task. A total of three recognition tasks were defined for our experiment referring to the concept of object type, object color and conformational responses. As introduced earlier, we trained individual error models for close and distant speech by processing the recorded speech samples offline

through our ASR system. The computed ASR error model for close speech is shown in Table 5.3, the ASR error model for distant speech in Table 5.4.

Semantic Concept	Correct Recognition		Confusion In-Domain		Confusion Out-Domain		Standard Deviation
<i>Type</i>	72/99	72.7%	1/99	1.0%	8/99	8.1%	0.12
<i>Color</i>	69/92	75.0%	2/92	2.2%	12/92	13.0%	0.18
<i>Confirm</i>	219/250	87.6%	4/250	1.6%	5/250	2.0%	0.12

Table 5.3 ASR error model for close speech trained on the Wizard-of-Oz dialog corpus

Semantic Concept	Correct Recognition		Confusion In-Domain		Confusion Out-Domain		Standard Deviation
<i>Type</i>	63/99	63.6%	4/99	4.0%	11/99	11.1%	0.13
<i>Color</i>	62/92	67.4%	2/92	2.2%	16/92	17.4%	0.14
<i>Confirm</i>	189/250	75.6%	4/250	1.6%	19/250	7.6%	0.11

Table 5.4 ASR error model for distant speech trained on the Wizard-of-Oz dialog corpus

How to read the table:

For the close speech case, the table states that in 75.0% of the cases a user uttered the information about the object's color such information was correctly understood by the ASR system. Further, in 1.6% of the cases where a conformational concept was uttered by the user, speech recognition caused a confusion of such semantic concept inside the same domain (e.g. the utterance of "yes" was understood as "no"). Furthermore, for the recognition rate of the semantic concept of object type a standard deviation of 0.12 was observed for different talkers.

5.3.3 3D pointing gesture recognizer error model

As introduced in chapter 5.2.4, a dedicated data collection experiment next to the Wizard-of-Oz experiment was performed for training the error model of the gesture recognition component. Based on 93 pointing gesture samples, we configured and trained the model on three features: the gesture recall rate, the average error angle, and the standard deviation of the error angle. It is important to note that hereby we always assumed fully precise gestures from the users towards the referenced objects. Correspondingly, any falsification within the tracked pointing direction of gestures was blamed upon the recognition component.

The collected data corpus showed that 74 of the 93 pointing gestures were correctly recognized by the system which leads to a recall rate of 79.5 %. The average observed error angle of the recognized gestures was 21.1 degrees with a standard deviation of 4.54. The relative high average error angle can be explained through the close proximity of the user to the referenced objects within our setup. Recall that the component identifies the direction vector of a pointing gesture by the line from the user's head to his pointing arm. Such technique shows to be an accurate solution for scenarios where the referenced objects are sufficiently far away from the pointing

user. However, given a short distance between the user and the referenced objects, such approach shows to be more error-prone and less accurate. More precise, for our setup, gestures tended to be tracked with a too steep orientation to the ground, with their direction vector mostly oriented below the table top. We compensated for that fact by allowing a generous maximal angle offset of 45 degrees of a tracked gesture to an object in order to consider such object still as a valid candidate.

Recall rate	Average error angle	Standard deviation Error angle
79.5%	21.1	4.54

Table 5.5 3D error model for pointing gesture recognizer

Figure 5.8 shows the overall distribution of the measured error angles. The illustrated bar graph within the figure hereby indicates the frequency of observed error angles within bins of an extent of 2.5 degrees. Further, each triangle within the graph indicates one concrete pointing gesture with its associated error angle. The figure shows that it is possible to moderately approximate the distribution of the error angles within our experiment setup by a normal distribution around the average error angle. We use such a condition within the simulation process in the following way: once a user action for a pointing gesture is triggered by the user model, a draw from the recall probability is performed. Assuming the simulated pointing gesture is assumed to be successfully detected, in the next step an error angle value is drawn from the shown normal distribution based on the observed average error angle of 21.1 and its deviation value of 4.54. The original simulated gesture direction vector, precisely pointing towards the desired object, is then falsified in compliance with the drawn error angle. Note that hereby the orientation of the direction falsification is picked randomly.

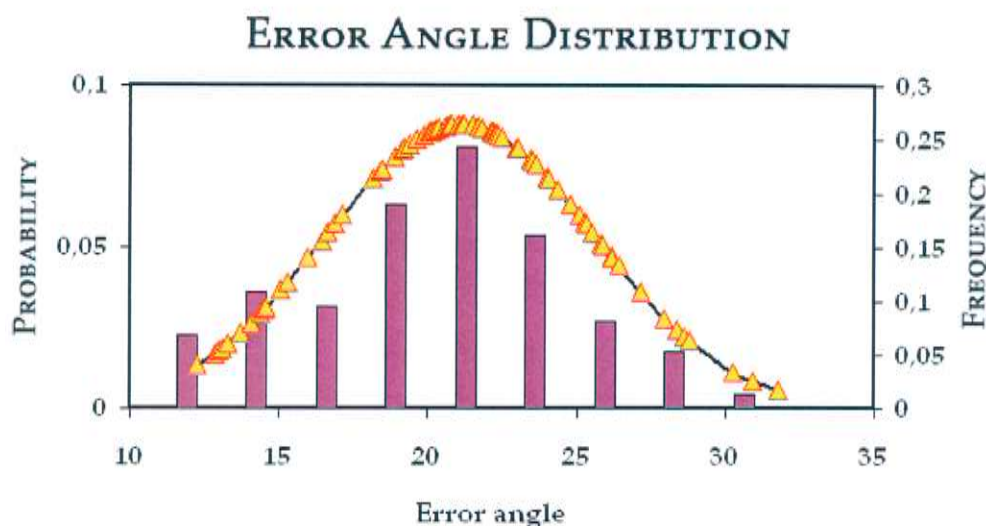


Figure 5.8 Normal distribution of observed error angles of the recognized gestures

5.4 The MDP design

Representing step three within our procedure model, after the training of the simulation model the design of the task-specific MDP was required. Such design involves to one part the action-state space configuration as well as to a second part the setup of the reward function. In the following, we will first introduce the designed reward function within our task and then present our action-state space design. We prefer such an order as our choice of the state features correlates with the designed reward function.

5.4.1 Reward function

The setup of our reward function was intended to facilitate the optimization of the dialog strategy towards three criteria: dialog success, dialog length and dialog naturalness. A key challenge hereby was to find a balanced representation of all three criteria within the reward function. For our scenario, the collected user feedback information introduced in chapter 5.2.3 was a valuable design criterion. As one goal, we intended to balance our optimization criteria in a way that the resulting average dialog length of our learned strategy would end up in the desired range of four to six system turns. Hereby we found the following reward configuration to be adequate for reaching such goal: every performed system action was penalized with a reward of -0.2. An unsuccessful dialog - i.e. the robot served the wrong item - was punished with a reward of -5, while a successful finish was rewarded by +1.0. Integrating the collected user feedback with respect to the naturalness of the dialog, we additionally penalized the direct repetition of the same system action also by -0.2⁹. Further, also a repeated prompt for a particular object attribute was penalized by -0.2. However, we also set a lower border of -0.4 for the reward to be received for every system action other than ending the dialog.

5.4.2 Action-state space design

The actions we designed within the MDP were equivalent to the actions available to the system within the dialog system. Correspondingly, a total of nine actions were defined in the way they are introduced in Table 5.1. Furthermore, following our earlier proposal in chapter 4.4, we used a rich set of eight state features (F1 to F8). Our selected state features, their possible values, as well as the motivation behind their choice will be now introduced in the following.

⁹ The direct repetition of the system action *ConfBest* was not penalized as we did not consider such system behavior as particularly unnatural but rather alike to the way a human might try to constrain the candidate set.

- **F1 – F3: Status of information slots**

[Possible Values (4): empty and never asked, empty and already asked, filled, confirmed]

Through the first three state features we incorporate the information about the status of the information slots object type, object color and object location within the state space. As introduced in chapter 2.3.1, encoding the status information of the information slots within the MDP is common practice. It facilitates strategy learning with respect to the fact which pieces of information has been collected so far and also with which level of confidence such knowledge is present. By using a dedicated state feature for each information slot, we facilitate learning of the optimal chronological order of slot-based prompts. We distinguish between the status “empty and never asked” and “empty and already asked” for such features as within our reward function we particularly penalize a repeated prompt for the same information slot. Only by incorporating the information if a prompt for a certain empty slot was executed before, the strategy will be able to optimize its behavior towards such criterion. As a side note, in case reliable recognition confidence scores are available from the ASR system, such scores also be used to further precise the slot status encoding (e.g. by splitting the status filled into “filled with high confidence” and “filled with low confidence”). However, for our ASR system we could not fallback on reliable confidence scores and thereby did not incorporate them into the state space.

- **F4: Number of candidates**

[Possible values (4): no candidates = 0, one candidate = 1, two to four candidates = 2, more than four candidates = 3]

The fourth state feature we use to model the dialog status is the number of current objects within the candidate list. Such feature comprises valuable information for proper strategy behavior. For example, given a high number of candidates, we would expect the optimal strategy to ask the user for more information in order to narrow the candidate pool. In the opposite case, given an empty set of candidates, the system can assume that recognition errors have occurred and should try to repair particular information by attempting to confirm obtained information. Finally, in the case where only one candidate exists, the system probably is well advised to spare the effort of additional questioning and instead, end the dialog soon.

- **F5: Speech condition**

[Possible values (2): close speech: CS, distant speech: DS]

As introduced earlier, within our experimental setup we consider the use of both, close speech and distant speech. Correspondingly, we incorporate a state feature indicating the present speech condition. The introduced trained ASR error models for speech conditions in Table 5.3 and Table 5.4 show a significant difference in recognition performance for close and distant speech. Through the incorporation of the speech condition state feature, we encourage the optimal strategy to adapt its behavior to such crucial environmental parameter. For example, in the case of using

distant speech we might accept that the optimal strategy will use more conformation requests than might be adequate in the case of close speech. Note that within performed simulation runs, we evenly distributed the use of close and distant speech and their associated error models from dialog to dialog.

- **F6: ASR history**

[Possible values (2): bad, good]

As a sixth state feature, we use a feature which refers to the history of the automatic speech recognition performance within a dialog. With the use of this feature, we intend to indicate the difficulty of speech recognition for a particular dialog interaction. The feature thereby correlates to some extent with the preceding speech condition feature. However, in contrast to the earlier case, the feature of ASR history is not static throughout a dialog interaction. Instead, it is designed to dynamically sense difficult conditions for speech recognition within a particular dialog. If we look back, we see that our trained ASR error models showed a significant standard deviation of recognition performance for different talkers. Precisely this is the phenomenon we want to address with the “ASR history” feature. Such feature should facilitate to adapt strategy behavior to dialog scenarios where talker or environmental conditions aggravate the speech recognition process. With the start of each dialog, the feature is set to good. Given the occurred condition that no user action at all could be processed from the user side as response to a system action, we set the feature to bad. The ASR history feature is also set to bad in cases where the confirmation of a received object attribute is neglected, as in such cases a high likelihood of the occurrence of earlier speech recognition errors is given. Similar as it is the case for the preceding feature, with the feature being set to bad, we would assume a change of the optimal strategy towards a safer strategy, represented by a more extensive use of conformational requests.

- **F7: Pointing gesture recognition condition**

[Possible values (2): bad, good]

In order to adequately address the gesture recognition component’s dynamics and sensitivity in the online-operation, we also designed a dedicated feature for such system component. In chapter 5.1.2.2 we briefly introduced the working mechanism of the used gesture recognition component. We mentioned that as a precondition of gesture recognition the user’s hands as well as his head have to be tracked correctly. Depending on the user’s body posture and movement, but also on other environmental factors such as light condition, such prerequisites for gesture recognition might be temporarily or for a longer period not accomplishable within a dialog interaction. By setting the state feature to bad for such conditions, the optimal dialog strategy is capable of adapting in a reasonable manner to such situations. For example, given the value of “bad” for the feature, we obviously would accept that the optimal strategy saves the effort of asking explicitly for a pointing gesture from the user until the critical condition resolves. As in the case of the speech condition

feature, within the dialog simulation we randomly set the gesture recognition condition feature for fifty percent to bad and in the other fifty percent to good. Unlike in the real experiment where the condition is observed before every system turn, the feature was considered static for the complete dialog interaction within the simulation process. Such deviation however should not influence the strategy learning process in any negative way.

- **F8: Last system action**

[Possible values (8): Greeting, AskType, AskColor, AskGesture, ConfType, ConfColor, ConfAllNI, ConfBest; see table 5.1 for description]

Last but not least, as a final state feature we also incorporated the information about the last system action within the state space. Considering the design of our reward function where the repetition of the same system action is explicitly penalized, the preceding system action provides necessary information for strategy optimization. However, considering the multiple possible values of such feature, it should not be kept secret that the incorporation of such historic information also leads to an immense expansion of the state space.

5.5 Strategy learning

Strategy learning was mainly performed using the presented approach in chapter 4.5. Within the simulation domain, we performed different kind of experiments. To one part we investigated the performance of different state space configurations as well as evaluated the performance of different λ values for the preferred Watkins' $Q(\lambda)$ algorithm. Further, we investigated the influence of the status of selected state features on the frequency of system actions within the optimal policy. The results of such mentioned experiments will be presented and discussed within chapter six.

5.6 Experimental validation

As already introduced in chapter 3.1.2, analyzing strategy performance within the simulation environment is a quick and inexpensive means of evaluation, but possesses only limited expressiveness. We have to consider that the strategy learned within the simulation is guaranteed to be optimal with respect to the built simulation model. However, an aspect that is hereby completely neglected is the uncertainty about how close the simulation model represents the real world task environment. Given a significant discrepancy between the simulation model and the real world scenario, a promising strategy performance could be observed within the simulation environment, while the same strategy might show very poor performance within the real world scenario. Therefore, especially when considering our use of a simulation model based on a relatively small data corpus and rather simplistic statistical methods, it is of utmost importance to validate the learned strategy's adequateness not only in the simulation domain but also in the real world scenario. Only through

such venture an expressive experimental validation of the suitability and strength of our procedure model can be reached.

Consequently, within the central experiment we carried out we evaluated the performance of our learned strategy within the simulation domain as well as in the real world experiment. Within this course we compared the performance of our learned strategy to the performance of a self-designed typical handcrafted strategy, which we considered as a challenging baseline system for our learned strategy. Hereby, the designed baseline strategy was designed in a way that it first attempts to collect the object type and color information. As soon as both slots are filled, a joint confirmation is executed. Subsequently, assuming that so far no pointing gesture of the user has been detected, the system action *AskGesture* is performed exactly once. Thereafter, the robot performs the system action *ConfBest* until the within that action referenced item is positively confirmed by the user. As a general guideline, the baseline strategy ends the dialog as soon only object remains within the candidate pool. For further exemplification, Listing B.1 within the Appendix shows two sample dialogs collected by the baseline strategy.

For the evaluation of both strategies within the simulation domain 10^5 artificially generated dialogs were analyzed. For our evaluation effort within the real world experiment a total of 18 test subjects were engaged of which only one test subject also participated within the Woz experiment. Hereby a total of 94 dialogs, 47 dialogs for each strategy, and 576 utterances were collected for evaluation. Equivalent to the Woz experiment, each participating user was asked to read through a prepared scenario description before the first dialog. The handout can be found in Appendix A. Due to the absence of a well performing online speech segmentation system for distant speech, all dialog interactions within the real world setup were carried out using head-mounted close talk microphones. Correspondingly, also the dialog interactions performed within the simulation were assuming the use of close speech in order to facilitate a better comparability of both domains. In order to address a potential learning effect of the user within the real world setup, we evenly switched between the use of the learned strategy and the baseline strategy within the first dialog of each user. Also we ensured not only that each user performed the same amount of dialog interactions for both strategies, but also that a certain item selection was always evaluated for both strategies in order to guarantee unbiased results. Chapter 6.2 will introduce the results of our experiment.

5.7 Simulation retraining

As a last experiment regarding the strategy learning process, we also performed the sixth step of our procedure model which corresponds to the retraining of the simulation model based on the online-operation experience. Hereby, such online-operation experience was represented by the real dialogs interactions performed within the context of the just introduced strategy evaluation experiment. Our purpose of this experiment was to analyze the extent to which the retraining process of the simulation model will influence the newly learned optimal strategy.

In order to retrain the simulation model, we transcribed the 94 collected online-operation dialogs in the same way as the collected Wizard-of-Oz corpus. Table 5.6 shows the resulting user model and Table 5.7 the close speech ASR error model trained solely on the collected online-operation experience.

Action	Type		Color		Gesture		Confirm	
<i>Greeting</i>	80/94	85.1%	47/94	50.0%	23/94	24.5%	0/94	0.0%
<i>AskType</i>	40/45	88.9%	4/45	8.9%	0/45	0.0%	0/45	0.0%
<i>AskColor</i>	0/59	0.0%	58/59	98.3%	0/59	0.0%	0/59	0.0%
<i>AskGesture</i>	0/65	0.0%	0/65	0.0%	65/65	100.0%	0/65	0.0%
<i>ConfType</i>	0/27	0.0%	0/27	0.0%	0/27	0.0%	27/27	100.0%
<i>ConfColor</i>	0/16	0.0%	0/16	0.0%	0/16	0.0%	16/16	100.0%
<i>ConfAllNI</i>	1/63	1.6%	2/63	3.2%	0/63	0.0%	61/63	96.8%
<i>ConfBest</i>	0/102	0.0%	0/102	0.0%	0/102	0.0%	101/102	99.0%

Table 5.6 User model trained on the online-operation dialog corpus

Semantic Concept	Correct Recognition		Confusion In-Domain		Confusion Out-Domain		Standard Deviation
<i>Type</i>	93/121	76.9%	1/121	0.8%	7/121	5.8%	0.17
<i>Color</i>	86/111	77.5%	6/111	5.4%	8/111	7.2%	0.17
<i>Confirm</i>	188/205	91.7%	1/205	0.5%	4/205	2.0%	0.10

Table 5.7 Close speech ASR error model trained on the online-operation dialog corpus

When comparing both models to the models which resulted from the Wizard-of-Oz experience (see chapter 5.3), we can observe a moderate degree of accordance. Clearly such similarity is positive news, as the observation attests to the fact that the environmental conditions of our performed WOz experiment obviously matched well the conditions present within the final experiment setup with respect to the user behavior and the ASR performance. Consequently, following our guidelines presented in chapter 4.7, we performed simulation retraining of the user model and ASR error model on a merged dialog corpus of the WOz experiment as well as the available online-operation experience. The resulting user models trained on the extended data corpus can be found in the Appendix in Table C.1 and Table C.2.

A completely different picture is demonstrated, however, when we compare the performance metrics of the gesture recognizer component within the online-operation and the earlier performed trial data collection (see chapter 5.2.4). The recall rate of performed pointing gestures within the online-operation was observed with a percentage of 52.2 %, while within the earlier performed dedicated data collection a significant higher recall rate of 79.5 % was observed.

The reason for such a significant discrepancy could be identified quickly: Within the initially performed trial experiment used to evaluate the gesture recognition performance, the test subjects were fully aware about the background and purpose of

the upcoming experiment. The system's prompt for pointing gestures was therefore expected and their execution was performed by the test subjects in a very thorough manner. As a matter of fact, during the experiment we noted for most test subjects that body and especially hand movements were completely isolated to the phases when a pointing gesture was demanded by the system. Such clearly ideal conditions for pointing gesture recognition however significantly changed within the online-operation. With the users not focusing solely on the performance of pointing gestures, but instead being involved within a task-oriented conversation, we observed a much higher activity level and more movement of the user, especially also of subconscious and incidental hand movements. Clearly, such condition aggravated the recognition of pointing gestures. Due to the presence of the additional fuzzy hand movements, spatial trajectories of real pointing gestures could be identified as such less frequently.

In order to avoid such discrepancies within the observed recognition performance, a better approach would have been to train the initial gesture recognizer error model not by an isolated data trial experiment, but instead on recorded video material of the Wizard-of-Oz dialog interactions. However, as introduced earlier, logistical reasons inhibited such proceeding within our case. In order to adapt our learned strategy to the occurred presented environmental changes, within the retraining of the simulation model we specified the recall rate of the gesture recognition component solely based on the online-operation corpus with 52.2%. With respect to the error model's parameters of the average error-angle and its standard deviation, we fell back on the training data from the WOz experiment. Due to a technical shortfall during the collection of the online-operation experience, we were not able to determine such values for the online-operation corpus. However, a significant change of such parameters was not to be expected.

6 Results and Discussion

In the following chapter we will describe and discuss the obtained results for the performed experiments which we already introduced within the preceding chapter. First, we will introduce the results of three experiments performed exclusively within the simulation domain. Subsequently, we will present our obtained results of the executed comparative study of our learned strategy and a handcrafted baseline strategy. Finally, we will discuss the observed effect of the performed simulation retraining process upon the learned optimal strategy.

6.1 Experiments within the simulation domain

6.1.1 Evaluating different state-space configurations

In our first experiment within the simulation environment, we evaluated the performance of different state space feature compositions. We started off with strategy learning from a state space configuration using only the introduced features F1 to F4. In the next run, we then added the additional state feature F5 and again learned the optimal strategy for this space configuration. We continued in that manner until all defined eight state space features were included within the state space configuration.

While we have so far promoted the use of Watkins' $Q(\lambda)$ method for strategy learning, in our experiment we drifted from his proposal. In order to facilitate a valid comparison of different state-space configurations, applying Watkins' $Q(\lambda)$ method would have required the use of a fixed λ value over all state-space configurations. However, as introduced earlier, the suitability of a particular λ value within Watkins' $Q(\lambda)$ approach strongly depends on the chosen state-space configuration. Therefore, using such approach for comparing the performance of different state-space configurations would have led to biased results. In order to avoid such effect,

we utilized simple Q-learning for strategy learning in the context of this learning experiment.

During learning, we followed a complete exploration strategy¹⁰. For each state-space configuration we performed a total of 10^7 training dialogs, which took approximately 45 minutes of computation on a Pentium IV machine with 1GB of memory. After such amount of dialog interactions, learning converged sufficiently for all evaluated state space configurations. Every 10^5 dialogs we evaluated the so far learned strategy by performing 10^5 dialogs using a greedy action selection. Within the evaluation runs, we evenly distributed the different settings for the speech condition and the gesture recognition condition. Figure 6.1 shows the observed learning curve of the different state space configurations with respect to the average collected reward.

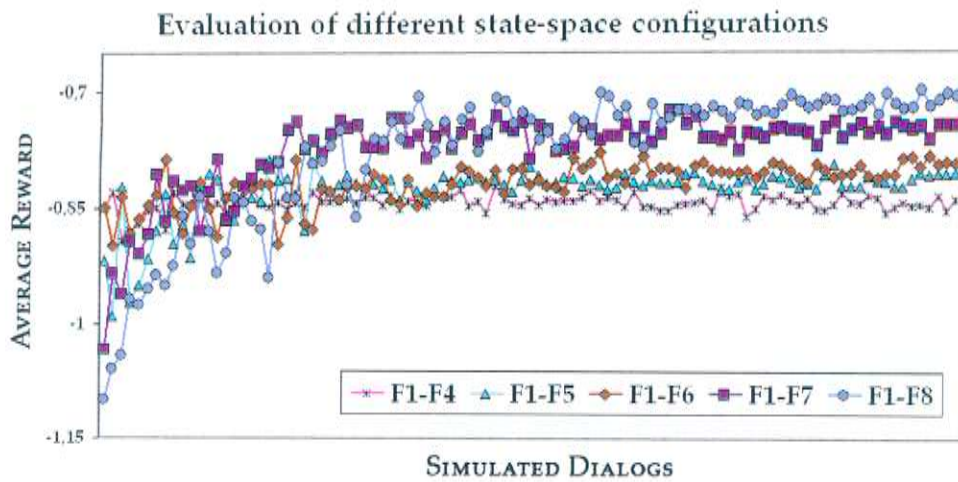


Figure 6.1 Strategy evaluation of different state-space configurations

Further, as illustrated in Table 6.1, the numeric performance results of the converged optimal strategies are listed with respect to the collected average reward, the average dialog length, the task success rate, as well as the number of visited states during learning.

Used features	Average reward	Average dialog length	Task success rate	Visited States
<i>F1-F4</i>	-0.846	5.531	86,91%	692
<i>F1-F5</i>	-0.797	5.341	88,01%	1105
<i>F1-F6</i>	-0.784	5.360	87,97%	2202
<i>F1-F7</i>	-0.740	5.270	88,92%	3494
<i>F1-F8</i>	-0.706	5.141	90,12%	14206

Table 6.1 Comparison of strategy performance using different state-space configurations

¹⁰ Every system action during the learning process was picked randomly.

As a central observation, the results show that the average collected reward and thereby the strategy quality increases with every added state feature. We also note that the average dialog length decreases with the amount of used state features while the likelihood for a successful task completion steadily increases. The conclusions inferred from such observations are multifaceted.

Firstly, the results confirm the adequateness and validity of each designed state feature. As the results empirically validate, every state feature introduces additional valuable information to the strategy learning process, which facilitates a more sophisticated and improved decision making of the learning agent in the simulation domain. In this context, the improving strategy performance with every state feature also validates that our designed simulation model is succeeding in modeling all the dialog dynamics addressed by the state features. For example, assuming we would have used only one joint ASR error model for close and distant speech, the addition of the corresponding speech condition state feature would not have resulted in any performance gain.

As a further contribution, the results also provide us an evaluation of the potential of single state features as well as their correlation with each other. As the attentive reader might have expected, the difference between the strategy performance of using state feature F1 to F5 and the strategy performance when using additionally the state feature of ASR history (F6) is rather marginal, but still leads to improvement. The reason for such behavior is based on the fact that a high correlation between feature F5 (speech condition) and feature F6 (ASR history) exists: Both features intend to indicate the level of difficulty for speech recognition within a particular dialog. Hereby, as an example, given a distant speech signal, the strategy might adapt by following a safer strategy by extensively using conformational responses. Assuming now also an indication of a bad ASR history, such information would further confirm the presence of a dialog interaction which is problematic with respect to speech recognition, however, would presumably only slightly further influence optimal strategy behavior. As a consequence, the observed performance gain when adding feature F6 to the state space is only small. Generally, a high impact of state features on strategy performance can always be observed when essentially new information is introduced to the state space. As an example, the addition of the feature referring to the gesture recognition condition (Feature F7) leads to a clear improvement for the optimal strategy performance despite the already rich state space. The reason for such effect is that the gesture condition feature comprises completely novel information not covered by the state features F1 to F6.

Finally we can declare that the results of this experiment support one of our earlier claims within this work: Also within a small-scale dialog interaction, as the one addressed within this work, a rich state space design is appropriate and also beneficial as the observed continuously improving strategy performance with every state feature shows.

6.1.2 Finding the best λ value

Having secured that the use of all designed state features leads to the significant best performing strategy, we examined in a second experiment the best matching λ value for Watkins' $Q(\lambda)$ method for such state-space configuration. Recall that we prefer using Watkins' $Q(\lambda)$ method over simple Q-learning, as the method allows a faster extensive population of the Q-values, as well as diminishes the effect of potential violations to the Markov property. Hence, the application of Watkins' $Q(\lambda)$ method is advisable especially when using an extensive state space. One shortcoming, depending on the chosen λ value and the state-space configuration, is that the use of Watkins' $Q(\lambda)$ method also includes the risk of a non-stabilizing learning process. Such phenomenon arises due to the method's simultaneous updates of several Q-values at one time. Precisely such a condition is the reason why an evaluation of different λ values for a given state-space configuration constitutes a valuable experiment in order to determine a parameter specification which allows a stable learning process of powerful strategies.

We performed five training sessions, setting λ to the values of 0, 0.2, 0.5, 0.8, and 1.0. Note that a λ value of zero is equivalent to Q-learning, while using a λ value of one corresponds to the use of a Monte Carlo method (see chapter 2.2.2.3.2). Again, we performed a total of 10^7 training dialogs, while we evaluated the strategies this time for better clarity every 10^6 dialogs, using 10^5 greedily performed dialog interactions. Figure 6.2 shows the resulting learning curves for the different values of λ .

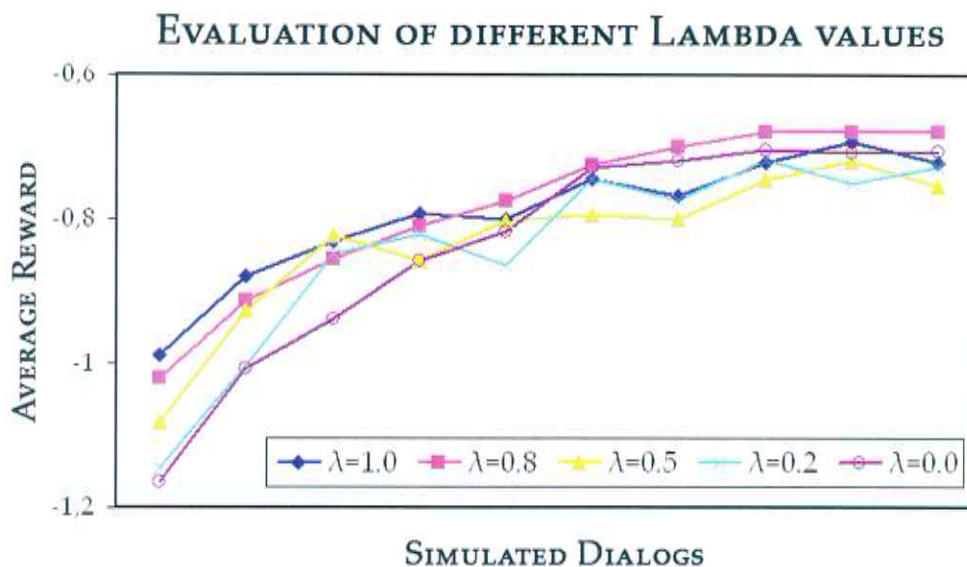


Figure 6.2 Evaluation of different Lambda values

As we would have expected, we note that in the very beginning of the learning process, the use of a big value for the Lambda parameter results in higher values for the observed average reward. When using Q-learning ($\lambda=0.0$), at this point of time within the learning process the action-state value table is still very sparse and an

adequate action selection is not yet possible. Within the other extreme, at the same point of time a Monte Carlo approach ($\lambda=1$) has performed already significantly more action-state value updates and the learned greedy strategy behavior matured already to a significantly better performance. The figure also demonstrates how a higher Lambda value leads usually to more fluctuations during the strategy learning process. For Q-learning, the learning graph appears monotonously rising until strategy convergence is reached. As for the other learning sessions, the learning process appears less stable. An exception to such general observation is the learning curve which is associated with the Lambda value of 0.8. As we can see, such parameter specification outperforms Q-learning with respect to the collected average reward while its associated learning curve also shows a stable condition. Therefore, such Lambda specification empirically qualifies as the optimal candidate for the given state-space configuration. Correspondingly, from this point forth, every time we refer to the optimal learned strategy, it can be assumed that such strategy was learned using Watkins' $Q(\lambda)$ method with λ being set to the value of 0.8.

6.1.3 State features and their effect on the optimal strategy

Having learned the optimal strategy performance within the simulation domain, we were interested in getting an insight into some identifiable general behavior patterns our optimal strategy adopted. Considering the rich state space of the policy, such an evaluation however can usually only be performed in a very coarse manner. One way to obtain cues regarding characteristic behavior of the learned strategy is to compute the strategy's frequency of system actions for a particular state feature and its possible values. We carried out such an analysis of the optimal policy for the state features "speech condition", "ASR history" and "gesture recognition condition".

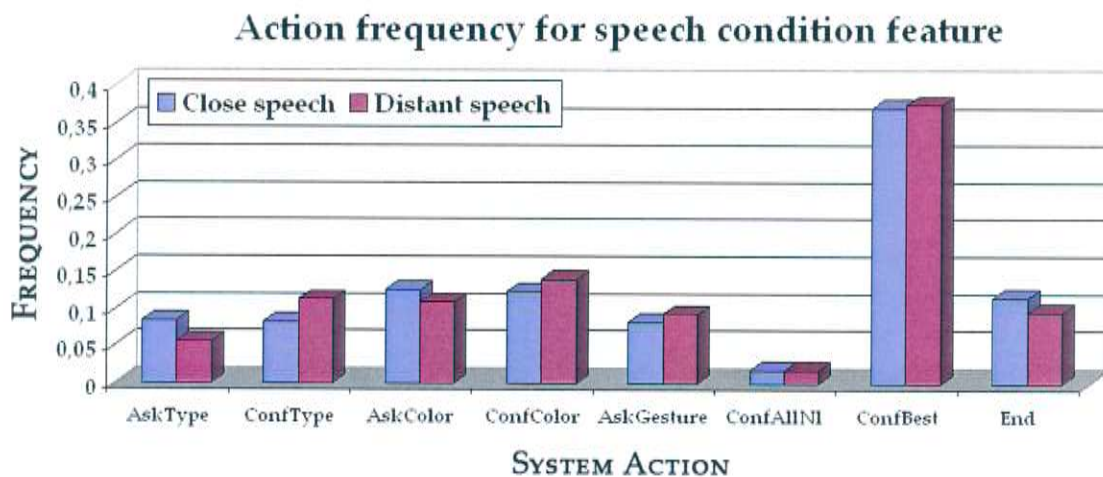


Figure 6.3 Action frequency within the optimal strategy for the speech condition feature

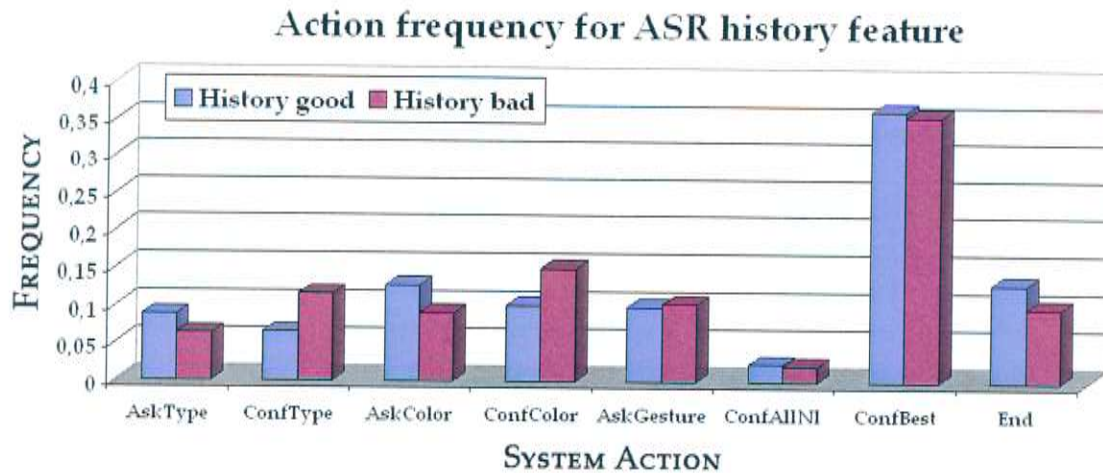


Figure 6.4 Action frequency within the optimal strategy for the ASR history feature

Figure 6.3 and Figure 6.4 show the system action frequencies of the optimal strategy for the state features “speech condition” and “ASR history” and their two possible values. As both features are used to indicate the difficulty of speech recognition for a particular dialog interaction, it is not surprising that the computed action distributions for their two values resemble. We can observe that given the status of distant speech or a bad ASR history, the strategy significantly increases the use of the system actions *ConfType* and *ConfColor*. Such effect illustrates how the strategy learns to intelligently adapt to particular challenging situations of speech recognition: within such conditions the strategy does not trust received information through the modality of speech as much as for the case of close speech or a good ASR history. Correspondingly, the optimal strategy prefers to confirm received information about object attributes over the modality of speech more frequently before ending the dialog interaction. Such behavior meets our earlier presented expectations about the features’ effect on an intelligent strategy behavior.

Furthermore, we also investigated the action frequency distribution for our modeled feature “gesture recognition condition”. Recall that in the simulation, we evenly distributed the occurrence of both conditions, while in the case of a bad condition simulated user gestures were assumed as unrecognizable by the gesture recognition component. Thus, the effect of the feature’s status on the optimal strategy is straightforward: Given a bad recognition condition, the system action to prompt the user for a pointing gesture towards the object (*AskGesture*) is hardly ever performed for any dialog state. Instead, the frequencies of system actions expecting user responses through the modality of natural language evenly increase.

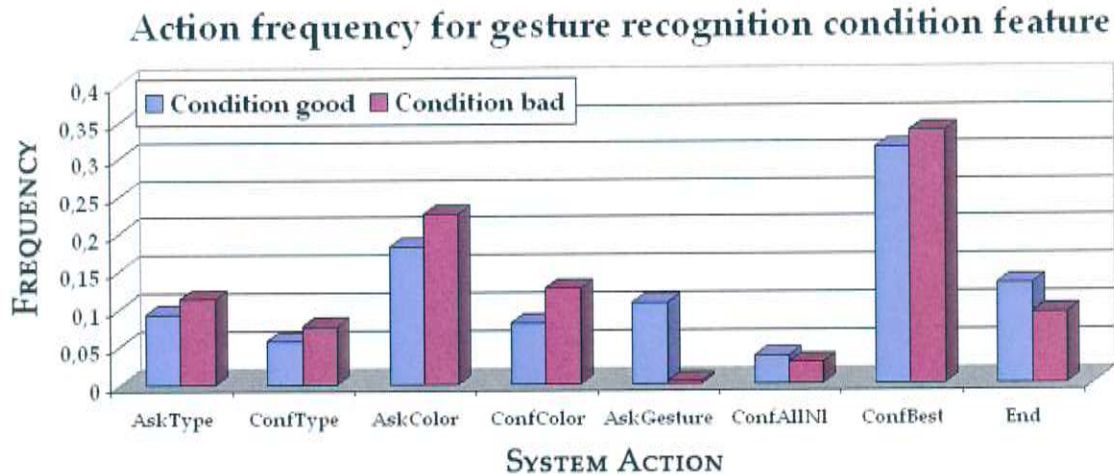


Figure 6.5 Action frequency within the optimal strategy for the gesture recognition condition feature

As a final comment, it is interesting to note that in all three presented figures, the system action to ask for the object color (*AskColor*) is significantly more present within the optimal strategy than the action to ask for the object type (*AskType*). As the complexity of both object attributes was designed in an equal extent with three possible values, such significant deviation might surprise at initial glance. However, if we take a look at our simulation model, we realize why the execution of *AskColor* seems more promising to the system than the action *AskType*. To one part the trained user model (Table 5.2) shows that for the action *AskColor* the requested information is actually provided with a higher likelihood than it is the case for the action *AskType*. Looking at the trained ASR error models (Table 5.3 and Table 5.4), we can further see that speech recognition performs more successful for the semantic concept of the object color than the semantic concept of the object type. Considering such two conditions, the preference of the optimal strategy towards the action *AskColor* is understandable and nicely demonstrates in which way the simulation model influences the behavior of the optimal strategy.

6.2 Experimental validation results

In the following, we will present and discuss our results of the performed comparative study of the learned strategy and the designed baseline strategy introduced earlier in subchapter 5.6. We evaluated the strategy performance for the simulation domain (SIM) on 10^5 artificially generated dialogs and for the real scenario (REAL) on 47 performed real dialog interactions for each strategy. Table 6.2 shows the results of our comparative study with respect to task success, the average collected reward, as well as the average dialog length.

SCENARIO	Baseline Strategy		Learned Strategy	
	REAL	SIM	REAL	SIM
<i>Dialogs</i>	47	10 ⁵	47	10 ⁵
<i>Successful tasks</i>	80,4%	83,3%	86,9%	91,3%
<i>Ø Reward</i>	-1.252	-1.067	-0.782	-0.688
<i>Ø Dialog length</i>	5.924	5.956	4.943	5.007

Table 6.2 Comparative study of the baseline strategy and the learned strategy

The key statement we can make from the listed results is that the performance of our learned strategy is clearly superior to the performance of the designed baseline strategy. Hereby, we can attest such supremacy not only in the simulation domain but, more preciously, also within the real world experiment. The predominance of the learned strategy is clearly observable with respect to all three evaluation criteria: the learned strategy receives in average higher reward values for the performed dialog interactions, leads to a higher percentage of successful task completion and executes in average one system turn less than the baseline strategy. Note that considering our design of the reward function (see chapter 5.4.1), hereby a higher collected reward value implicitly also indicates a higher degree of naturalness of the dialog interactions.

To provide a closer insight within the caused dialog interactions of both strategies for the real scenario, Appendix B presents exemplary dialogs for both strategies. For the case of the learned strategy we also list the learned state-action values for the performed optimal action as well as the learned second and third best system action. During the analysis of the dialogs collected within the real scenario, we observed that for nearly all failed dialog interactions using the baseline strategy the maximum number of ten dialog turns was reached and therefore the dialog was prematurely ended. Hereby, such failed interactions always occurred within scenarios where speech recognition was especially problematic. In such causes, the baseline strategy had to invest already several turns solely to obtain and confirm the speech-based information of the object type and object color. More intelligently, in such cases the learned strategy preferred asking for a pointing gesture and then focused on a selection procedure less prone to speech recognition errors by confirming candidate objects through the repeated execution of the *ConfBest* system action. On the other side, failed interactions caused by the learned strategy resulted mostly from misrecognition of speech inputs and falsely detected pointing gestures in situations where the state features did not succeed in indicating particularly error-prone recognition conditions. In these scenarios, the learned strategy did not extensively pursue the confirmation of particular information slots but relied on the received information and strived for a fast dialog closure. Such strategy behavior then resulted for a small number of dialogs within a failed task completion.

An interesting behavior pattern we could observe for the learned strategy during online-operation can uncouthly be described as a “lucky shot” strategy. Such action behavior occurred mostly when in response to the mixed-initiative greeting action a pointing gesture of the user was recognized. In such cases, independent of the number of detected valid candidates, the strategy performed as a next action the *ConfBest* system action. Obviously, hereby the system speculated that the perceived pointing gesture was recognized in such an accurate manner, that the best matching candidate, and thereby the referenced object through the *ConfBest* action, represented directly the desired object of the user. Assuming the fulfillment of such speculation, the strategy reached a successful and very short dialog interaction. For the opposite case, and considering a high number of candidates, the strategy continued by first collecting further information about the object’s attributes before (re)starting to specifically confirm concrete objects through the *ConfBest* action. For further exemplification, the second dialog within Listing B.3 shows an occurred interaction during online-operation where such strategy behavior led to an early successful closure of the dialog.

A further valuable observation we could derive from the obtained results includes the comparison of the performance of both strategies within the simulation domain and the real world scenario. Despite the significant difference within the number of performed dialogs for both domains, we can observe a moderate resemblance of the computed strategy performance metrics for both domains. Such observation is a positive result as it implies a moderate accuracy and suitability of our simulation model. However, while the average dialog length for both strategies and scenarios is nearly identical, we see that for both strategies the successful task completion rate and the collected average reward are slightly higher within the simulation domain. In fact, we expected such kind of discrepancy within the results. In chapter 5.7, we mentioned the weaker performance of our gesture recognition component within the real world scenario compared to the trained parameters for the simulation model. Correspondingly, it is only logical that the performance of both strategies slightly worsens within the real world scenario, considering the - compared to the simulation model’s assumptions - impaired system performance. However, the direct comparability of strategy performance within the simulation and real world scenario suffers also from a further aspect: Within the dialog simulation process, we switch for every artificial dialog interaction the value of the gesture recognition condition from good to bad and vice versa for the complete dialog interaction. Such simulated distribution of the gesture recognition feature however only represents a simplistic and very coarse approximation of the real distribution. However, due to the gesture tracking component’s absence within the WOz experiment, we were unable to learn such distribution within a dialog context in a data-driven way. Yet, such an approach would have facilitated a more accurate simulation model and ultimately presumably result in less deviation of strategy performance results within the simulation domain and the real world experiment.

However, if we refocus on our principal motivation within this work, our key objective is not to build a simulation model that necessarily has to match the

simulation model one-to-one. Instead, a more important demand we impose on the simulation model is that it provide us with artificial dialog experience from which we can learn complex dialog strategies, capable of outperforming typical handcrafted strategies. The results presented in Table 6.2 empirically validate that our proposed procedure model is principally capable to reach such goal. Hereby it has to be mentioned that the expressiveness of the demonstrated results is restricted by two conditions: to one part by the quality of the baseline strategy, and to a second part by the limited amount of dialogs on which we evaluated the performance of the strategies within the real dialog scenario. With respect to our baseline strategy, a determination of its adequateness as challenging and expressive baseline system is obviously only possible by subjective means. However, we are convinced of the fact that the quality of our handcrafted strategy does justice to the average quality level of common manually designed dialog strategies, and thereby such strategy indeed represents an appropriate baseline system within the context of this work.

Regarding the amount of collected real dialogs used for strategy evaluation, clearly a higher number would have been desirable and would further manifest the validity of our approach. Yet, the expense of such sort of data collection did not allow a bigger evaluation corpus within the scope of our work. However, the moderate resemblance of the observed real world strategy performance to the associated performance values of the simulation model (which were evaluated on many more dialogs) at least foreshadows a moderate accuracy of our obtained values. Additionally, we also performed a statistical T-Test on the observed average reward value within the real experiment for both strategies. Positively, the test showed that we can dismiss the null hypothesis, i.e. there is no statistically significant difference within the performance of the learned and the baseline strategy, with an error probability of only 2,8%.

Finally, independent of any numerical results or statistical evaluations, the analysis of the performed dialog interactions using the learned strategy exposed highly intelligent behavior. In all probability, such observed complex strategy behavior would not have been ascertained within a handcrafted strategy design. This observation alone already suggests the high potential and utility of our introduced approach to strategy learning.

6.3 Simulation retraining results

The final experiment we performed involved the retraining of our simulation model based on the online-experience we obtained through the previously performed strategy evaluation experiment. In chapter 5.7, we introduced in detail the data we used for the simulation retraining process as well as our motivation behind such decisions. Appendix C provides a numerical description of all three used simulation model components. The purpose of the experiment was to observe the extent of changes the simulation retraining process caused to the learned strategy. In order to perform such comparison, we fell back on a technique already introduced earlier. We computed the frequencies of all system actions within the learned strategy to analyze

its overall behavior. Figure 6.6 shows the results of such computation for the strategy learned solely from the WOz corpus and the strategy learned from the extended data corpus, including online-operation experience. Note that within the learning process we only considered close speech interactions.

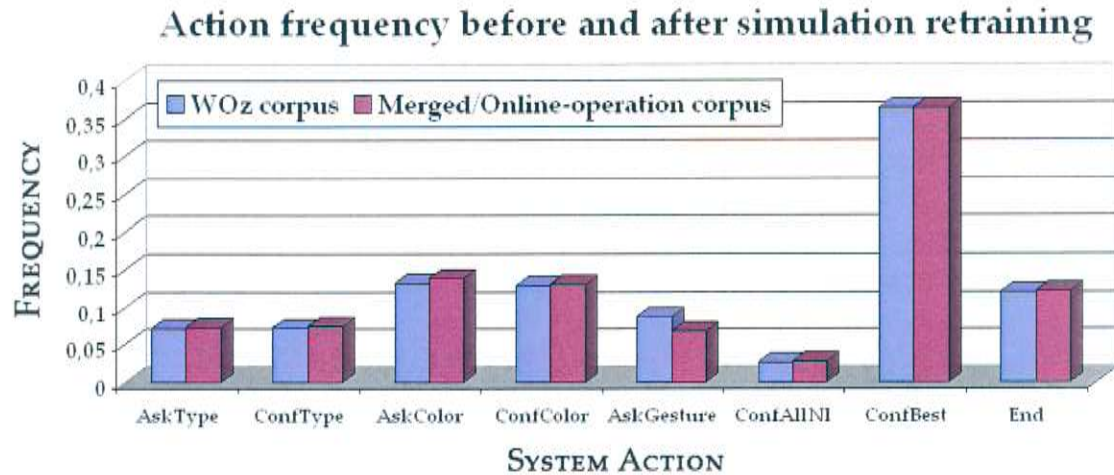


Figure 6.6 Action frequency of the optimal strategy before and after simulation retraining

The illustrated results show that the distribution of the action frequencies for both learned strategies is very similar. Such a condition by itself, however, does not allow the conclusion that both strategies also act in the same way for particular dialog states. For such determination, we further evaluated the number of visited states (not action-state pairs) for which the learned optimal action differed for both strategies. Our analysis showed that only in 53 of 1924 states (2.7 %) the optimal action for a visited state actually changed through the simulation retraining. As the above figure already implies, hereby for the big majority of changes the execution of the system action *AskGesture* was replaced by another optimal action. Clearly, such kind of strategy adaptation was to be expected, considering that the main alteration to the simulation model referred to a weaker performance of the gesture recognition component within the real world scenario.

The experiment shows that principally the methodology of simulation retraining based on online experience is an adequate and advisable step to adapt the learned strategy to occurring environmental changes during online-operation. Further, the fact that the aggregation of additional online-operation experience only led to minor changes within the optimal strategy behavior empirically validates our claim that for the addressed types of dialogs already a small Wizard-of-Oz dialog corpus is capable of providing a fair approximation of the, with respect to the used simulation framework, optimal strategy. Clearly, such observation is principally restrained to dialog scenarios similar to the one performed with respect to dialog complexity. Given a more extensive dialog scenario (e.g. additional information slots or modalities), we can infer that also more dialogs would have to be collected within the Wizard-of-Oz experiment in order to reach similar results.

7 Conclusion and further work

7.1 Summary and conclusion

This thesis introduced a holistic approach to rapid automatic dialog strategy learning, particularly tailored for typical human-robot interactions. Such an approach is generally highly valuable with respect to two aspects: Firstly, it significantly accelerates the overall development process of dialog systems. And secondly, such approach provides a high quality, at best optimal, automatic design of the crucial system's dialog strategy.

As a key contribution of our approach, we relaxed a central restriction of earlier work performed within the same domain: To date, automatic strategy learning has been applied to dialog systems which have been already deployed to the real world using handcrafted strategies. In such work, strategy learning was then performed based on already present extensive online-operation experience. In contrast to such proceeding, our more pragmatic approach showed to facilitate rapid automatic strategy learning concurrently with the development process of dialog systems. Thereby, optimized learned strategies are available from the first moment of online-operation and also tedious handcrafting of dialog strategies is fully omitted. As a clear novel contribution, this work showed to perform automatic learning of not only solely speech-based, but multimodal dialog strategies.

Our approach is primarily based on the argumentation that for the addressed type of dialogs, already from a relatively small dialog corpus a dialog simulation model can be trained, which is capable of providing sufficiently accurate artificial dialog experience to facilitate automatic learning of intelligent and powerful dialog strategies. Hereby, our simulation model is particularly tailored for the within human-robot interaction typical slot-based, task-oriented and multimodal dialog interactions. For the training of the simulation model, we use an inexpensive Wizard-Of-Oz study as sole instrument for data collection. Through the application of

simplistic but expressive probabilistic models within dialog simulation we challenge data sparsity issues.

In order to empirically validate our approach, we applied the proposed procedure model to a concrete human-robot dialog interaction. Subsequently to the strategy learning process, we compared the performance of the learned dialog strategy to the performance of a typical handcrafted baseline strategy within the simulation domain and the real world scenario. The evaluation of the obtained results showed that the learned strategy was capable of significantly outperforming the non-trivial handcrafted strategy with respect to an optimization function which included the criteria of the success, length and naturalness of dialog interactions. Further, we were able to detect and expose intelligent behavior patterns the learned strategy adopted. Due to the use of well-thought and sophisticated state space design, the learned strategy also showed to be capable of wisely adapting to dynamic environmental conditions present within our concrete dialog scenario. Considering that the complete simulation training and thereby strategy learning process was based on only 82 dialogs collected within a Wizard-Of-Oz experiment, the learned strategy's meritorious performance clearly implies the validity of our taken approach and the proposed procedure model to strategy learning.

A valuable practical lesson we learned during the experimental application of our procedure model was that it is advisable to collect any kind of data utilized within the simulation training process only within the full context of the dialog interaction. We experienced that isolated data collection experiments easily lead to environmental conditions which are inconsistent with the environment of the final task setup. Ultimately, the incorporation of such collected data within the training process of the simulation model threatens the optimality of the learned strategy within the real world scenario. Correspondingly, within the concrete context of our procedure model, we underlined our proposal of collecting all information about user and system performance at best within one single Wizard-of-Oz experiment closely designed to the anticipated final system setup.

We are aware of the fact that all performed empirical validations of our approach and procedure model are restricted with respect to their expressiveness to dialog scenarios similar to the performed experiment. Further applications of our approach to discriminative dialog interaction setups, e.g. scenarios using a greater information space or other modalities, will need to show the universal applicability of our approach within task-oriented human-robot interactions. Nevertheless, the obtained results within this thesis represent an informative and promising first insight into our approach's practical applicability as well as its overall potential.

7.2 Further work

Within the following final subchapter, we want to provide suggestions for further work which we consider as particularly interesting within the context of this introduced thesis.

Hereby, one domain of further work we consider worthwhile refers to the process of data collection within simulation-based strategy learning. Within our procedure model, the proposed data collection for the training of the simulation model was isolated to the execution of a Wizard-of-Oz experiment. As a positive consequence of such approach, the collected experience is always fully task-based and should thereby lead to an accurate simulation model. However, it is also imaginable that, especially with respect to error modeling of system components, task-independent data could be incorporated within the simulation training process. As an example, the ASR error model for a task might be predictable to a certain level based on experienced performance of the same speech recognition system in earlier application scenarios. While such data ascertainment clearly could not provide the same accuracy as a data collection performed within the concrete task scenario, it could be used as fallback option within situations of data sparsity. Such task-independent data could be used within the initial iteration of strategy learning, before the simulation model could be later refined based on online-operation experience. The development of techniques for incorporating and exploiting task-independent data within the training process of the dialog simulation represents a so far nearly untouched, but very interesting research field within the domain of automatic strategy learning.


Another field of future work we consider refers again to the domain of dialog simulation. Within our proposed procedure model, we consider the modeled bi-gram user model as the weakest part within our ambition of building an accurate but still low-conditioned simulation model. Assuming a longer online-operation period of a particular dialog system, the data corpus utilizable for the training of the dialog simulation steadily increases. At some point, the extent of experience would suffice to train a more complex user model. Just as one example, the so far used bi-gram model could then be extended to a tri-gram model, i.e. user actions are conditioned on the last two system actions. It would be interesting to observe the effect of the use of different more complex user models on the performance of the learned strategy within the real world scenario. Assuming a significant change to strategy performance would be present, such evaluation could provide valuable cues towards promising additional parameters to be incorporated within the simulation of user behavior.

Another proposal for further work is one that applies to nearly all work performed within the domain of automatic strategy learning: the addition and evaluation of further promising state features within the state space. Earlier, we already mentioned the imaginable valuable use of ASR confidence scores within the state space. However, many other additional state features are conceivable to improve the

quality of the learned strategy. As one further example, the perceived emotional status of the user could represent an interesting parameter within strategy optimization. In this context, Holzapfel et al. [41] proposed a framework for using such emotional cues in a dialog system. Note that within simulation-driven strategy learning, the addition of the user's emotional status to the state space is only then reasonable if also the user model is capable of simulating user actions adequately conditioned on such parameter.

Last but not least, a final domain of further work we consider refers to the concrete human-robot dialog scenario we setup within this thesis: It is planned to port the developed bartender dialog scenario to the earlier introduced robot platform ARMAR of the SFB588 project. Capitalizing on the advanced capabilities of such system, the robot's action of pointing towards particular objects on the table as well as actually serving the selected object at the end of a dialog is intended to be realized using the robot's anthropomorphic arm system. Further, when using the ARMAR system, also the automatic recognition of the objects' locations as well as their classification with respect to color and type at the beginning of each dialog is aspired using a vision system developed by Azad et al. [42]. Within this context, it would represent an interesting extension to this work to incorporate potential uncertainties of the vision system, e.g. about the existence or attributes of the objects, to the learning process of the dialog strategy.

A Handouts




Interactive Systems Laboratories
interACT
research

Thesis, Thomas Prommer
Carnegie Mellon University,
Pittsburgh, 11/23/2005

Scenario

Description



Dear test user,

in the upcoming experiment you will get to know to our bartender robot Robbie – in this experiment partly simulated by a human.

Your goal will be to make Robbie serve you a particular item of your choice from the table setting.

Therefore Robbie will initiate a multi-modal dialog. Note that Robbie will be listening as well as watching you.

Several dialog runs will be performed in a row. Before each dialog start, we would like you to silently pick a different item from the table which you would like Robbie to serve you.

Within a dialog, questions should be answered consistent to your item choice and no change to your selection should be done.

After each dialog we will ask you for your experienced naturalness, length adequacy and success of the dialog (terms will be explained by the time we ask you the first time for it).

Any questions? Don't hesitate to ask them! But if it's all clear to you then ...

**... Let's get started
and Good Luck!**

Listing A.1 Handout scenario description for Wizard-of-Oz experiment

Dialog Evaluation

Name: _____

Time: ____:____ am / pm

Dialog number	Naturalness	Length adequacy	Successful
#1	-2 -1 0 +1 +2	-2 -1 0 +1 +2	Yes / No
#2	-2 -1 0 +1 +2	-2 -1 0 +1 +2	Yes / No
#3	-2 -1 0 +1 +2	-2 -1 0 +1 +2	Yes / No
#4	-2 -1 0 +1 +2	-2 -1 0 +1 +2	Yes / No
#5	-2 -1 0 +1 +2	-2 -1 0 +1 +2	Yes / No
#6	-2 -1 0 +1 +2	-2 -1 0 +1 +2	Yes / No
#7	-2 -1 0 +1 +2	-2 -1 0 +1 +2	Yes / No
#8	-2 -1 0 +1 +2	-2 -1 0 +1 +2	Yes / No

(please circle your responses)

Naturalness: How natural did you experience the dialog communication with the robot? -2 (*very artificial*), 0 (*moderate*), +2 (*very natural*),

Length adequacy: Did the length of the dialog meet your expectations for the particular task? *dialogue was longer than expected* (-2), *dialogue length met expectation*(0), *dialog was longer than expected* (+2)

Success: Did the robot serve you the correct item?

**Two final questions ...**

How did you feel about the incorporation of gestures in the dialogue?

-2 -1 0 +1 +2
Doesn't make sense. **It's ok.** **It is beneficial.**

How natural does it feel being asked for a gesture compared to being asked for a speech input e.g. for a color or object ... (just circle your opinion)

-2 -1 0 +1 +2
more artificial **equally natural** **more natural**

Thank you for your participation !

And please feel free to add any comments on the back side!

Listing A.2 Survey template for user interviews coupled to WOz experiment



Thesis work, Thomas Prommer
University of Karlsruhe,
Karlsruhe, 03/14/2006

Scenario

Description



Dear test user,

In the upcoming experiment you will get to know to our bartender robot Robbie.

Your goal will be to make Robbie serve you a particular item of your choice from the table setting.

Therefore Robbie will initiate a multimodal dialog. Note that Robbie will be listening as well as watching you.

Several dialog runs will be performed in a row. Before each dialog start, we would ask you to pick a different item from the table and communicate your choice to us.

Within a dialog, questions should be answered consistent to your item choice and no change to your selection should be done.

Any questions? Don't hesitate to ask them! But if it's all clear to you then ...

**... Let's get started
and Good Luck!**

Listing A.3 Handout scenario description for validation experiment

B Sample dialogs

B.1 Wizard-of-Oz dialog example

TTS System: "Hello my name is ROBBI. Which item can I serve you?"
User: "It's a blue cup."
Wizard: *Wizard simulates a recognition error by providing the information "blue plate" to the system and chooses next action.*

TTS System: "Did you say the item is a plate?"
User: "No it isn't."
Wizard: *Wizard relaxes information of the object type and chooses next action.*

TTS System: "Did you say the item is blue?"
User: "Yes it is."
Wizard: *Wizard confirms the object color and chooses next action.*

TTS System: "Please point towards your desired item."
User: User points towards the item.
Wizard: *Wizard provides an observed gesture direction to the system (level of accuracy limited to left, center and right) and chooses next action.*

TTS System: "What type has the item?"
User: "It is a cup."
Wizard: *Wizard provides correct object type information to the system.*

TTS System: "Is them item I am pointing at the one you wish for?"
Hum. Robot: *Points at a particular blue cup on the table following a received command by the wizard.*
User: "Yes it is."
Wizard: *Wizard confirms the referenced object and ends the dialog.*

TTS System: "Thank you for your patience. I will serve your item now."

Hum. Robot: *Serves a particular blue cup from the table to the user following a received command by the wizard.*

Listing B.1 Sample dialog interaction of the performed Wizard-of-Oz experiment

B.2 Dialog examples of online-operation using the baseline strategy

Robot: "Hello my name is ROBBI. Which item can I serve you?"
 User: "The blue plate over there." *User points towards desired item #17.*
 ASR System: "Plate over there."
 Gesture Rec.: *Component successfully recognizes gesture.*

Robot: "Which color has your item?"
 User: "It's blue."
 ASR System: "It's blue."

Robot: "Did you say your desired item is a blue plate?"
 User: "Yes."
 ASR System: "Yes."

Robot: "Is the item I am pointing at the one you wish for?" *Points at object #20.*
 User: "No."
 ASR System: "No."

Robot: "Is the item I am pointing at the one you wish for?" *Points at object #17.*
 User: "Yes."
 ASR System: "Yes."

Robot: "Thank you for your patience. I will serve your item now." *Robot serves correct item #17.*

Total reward: +0.0

Robot: "Hello my name is ROBBI. Which item can I serve you?"
 User: "Give me a bottle."
 ASR System: "Give me a bottle."

Robot: "Which color has your item?"
 User: "It's blue."
 ASR System: "It's blue."

Robot: "Did you say your desired item is a blue bottle?"
 User: "Yes."
 ASR System: "Yes."

Robot: "Please point towards the item."
 User: "It's this bottle." *User points towards the item #2.*
 ASR System: "It's this one."
 Gesture Rec.: *Component successfully recognizes gesture.*

Robot: "Is the item I am pointing at the one you wish for?" *Points at object #2.*
 User: "Yes."

ASR System:	"Yes."
Robot:	"Thank you for your patience. I will serve your item now." <i>Robot serves correct item #22.</i>
Total reward: +0.0	
Robot:	"Hello my name is ROBBI. Which item can I serve you?"
User:	"I like something to drink."
ASR System:	"I like to have the red cup."
Robot:	"Did you say your desired item is a red cup?"
User:	"No."
ASR System:	"No."
Robot:	"What type has your desired item?"
User:	"A glass."
ASR System:	"A glass."
Robot:	"What color has your desired item?"
User:	"Blue."
ASR System:	"Bottle."
Robot:	"Sorry, I have to ask you again what color has your desired item?"
User:	"The color it's blue."
ASR System:	"That cup in blue."
Robot:	"Did you say the item you wish for is a blue cup?"
User:	"I don't understand."
ASR System:	"This thing."
Robot:	"Did you say the item you wish for is a blue cup?"
User:	"No, sorry."
ASR System:	"No."
Robot:	"Sorry, I have to ask you again. What type has your desired item?"
User:	"It's a drink."
ASR System:	"It's a thing."
Robot:	"What type has your desired item?"
User:	"It's a glass."
ASR System:	"It's a glass."
Robot:	Ends dialog due to a maximum of ten user interactions. "Thank you for your patience. I will serve your item now." Robot serves wrong item #20.
Total reward: -7.6	

Listing B.2 Two sample dialog interactions collected with the final experiment using the baseline strategy

B.3 Dialog examples of online-operation using the learned strategy

Format of the state information:

Status Object Type – Status Object Color – Status Object Location – Candidates – Speech Condition – ASR History – Gesture Recognition Condition – Last System Action

Robot: "Hello my name is ROBBI. Which item can I serve you?"
 User: "The blue bottle."
 ASR System: "Bottle."
 State: Filled-EmptyNotAsked-EmptyNotAsked-2-CS-Good-Good-Greeting
 Best Actions: **AskGesture -0.307**; ConfType -0.426; AskColor -0.474

Robot: "Could you please point towards your desired item?"
 User: *User points towards object #12.*
 Gesture Rec.: *Component successfully recognizes gesture.*
 State: Filled-EmptyNotAsked-Filled-1-CS-Good-Good-AskGesture
 Best Actions: **ConfType 0.124**; ConfBest 0.003; End -0.003

Robot: "Did you say your desired item is a bottle?"
 User: "Yes."
 ASR System: "Yes."
 State: Confirmed-EmptyNotAsked-Filled-1-CS-Good-Good-ConfType
 Best Actions: **End 0.863**; ConfBest 0.594; AskColor 0.419

Robot: "Thank you for your patience. I will serve your item now. *Robot serves correct item #12.*

Total reward: +0.4

Robot: "Hello my name is ROBBI. Which item can I serve you?"
 User: "That blue cup." *User points at item #14.*
 ASR System: Fails to detect speech input.
 Gesture Rec.: Successfully detects gesture.
 State: EmptyNotAsked-EmptyNotAsked-Filled-2-CS-Good-Good-Greeting
 Best Actions: **ConfBest 0.357**; AskType -0.046; AskColor -0.087

Robot: "Is the item I am pointing at the one you wish for?" *Points at item #14.*
 User: "Yes it is."
 ASR System: "Yes it is."

Robot: "Thank you for your patience. I will serve your item now." *Robot serves correct item #14.*

Total reward: +0.6

Robot: "Hello my name is ROBBI. Which item can I serve you?"

User: "I would like to have the yellow cup over there." *User points towards item #5.*

ASR System: "I'd like that yellow cup."

Gesture Rec.: *Component successfully recognizes gesture.*

State: Filled-Filled-Filled-0-CS-Good-Good-Greeting

Best Actions: **ConfType -0.661**; ConfColor -0.768; ConfBest -0.949

Robot: "Did you say your desired item is a cup?"

User: "Yes."

ASR System: System fails to detect speech input.

State: Filled-Filled-Filled-0-CS-Bad-Good-ConfType

Best Actions: **ConfColor -1.244**; ConfBest -1.279; ConfType -1.285

Robot: "Did you say your desired item is yellow?"

User: "Yes."

ASR System: "Yes."

State: Filled-Confirmed-Filled-0-CS-Bad-Good-ConfColor

Best Actions: **ConfType -0.581**; ConfBest -1.090; End -2.771

Robot: "I have to ask you again, did you say your desired item is a cup?"

User: "Yes."

ASR System: System fails to detect speech input.

State: Filled-Confirmed-Filled-0-CS-Bad-Good-ConfType

Best Actions: **ConfBest -1.009**; ConfType -1.184; End -2.885

Robot: "Is the item I am pointing at the one you wish for?" *Robot points towards item #6.*

User: "No, to the left."

ASR System: "No that."

State: Filled-Confirmed-Filled-0-CS-Bad-Good-ConfBest

Best Actions: **ConfType -0.954**; ConfBest -1.144; End -2.760

Robot: "Did you say your desired item is a cup?"

User: "Yes."

ASR System: "Yes."

State: Confirmed-Confirmed-Filled-0-CS-Bad-Good-ConfType

Best Actions: **ConfBest -1.601**; End -3.03

Robot: "Is the item I am pointing at the one you wish for?" *Robot points towards item #5.*

User: "Yes."

ASR System: "Yes."

Robot: "Thank you for your patience. I will serve your item now." *Robot serves correct item #5.*

Total reward: -0.4

Robot: "Did you say your desired item is a cup?"

User: "Please hand me the blue bottle."

ASR System: "Please give me the blue bottle."

Gesture Rec.: *Component confuses user movement as pointing gesture.*

State: Filled-Filled-Filled-1-CS-Good-Good-Greeting

Best Actions: **End 0.809**; ConfAll 0.559; ConfColor 0.496

Robot: "Thank you for your patience. I will serve your item now." *Robot serves wrong item #16.*

Total reward: -5.2

Listing B.3 Four sample dialog interactions collected within the final experiment using the learned strategy

C Simulation retraining

Action	Type		Color		Gesture		Confirm	
<i>Greeting</i>	147/176	83.5%	98/176	55.7%	44/176	25.0%	0/176	0.0%
<i>AskType</i>	70/80	87.5%	6/80	7.5%	1/80	1.3%	0/80	0.0%
<i>AskColor</i>	1/101	1.0%	96/101	95.0%	0/101	0.0%	0/101	0.0%
<i>AskGesture</i>	0/125	0.0%	0/125	0.0%	124/125	99.2%	0/125	0.0%
<i>ConfType</i>	0/92	0.0%	0/92	0.0%	0/92	0.0%	92/92	100%
<i>ConfColor</i>	0/78	0.0%	1/78	1.3%	0/78	0.0%	78/78	100%
<i>ConfAllNI</i>	1/84	1.2%	2/84	2.4%	0/84	0.0%	82/84	97.6%
<i>ConfBest</i>	1/204	0.5%	0/204	0.0%	0/204	0.0%	203/204	100%

Table C.1 User model trained on merged dialog corpus of WOz experiment and online-operation corpus

Semantic Concept	Correct Recognition		Confusion In-Domain		Confusion Out-Domain		Standard Deviation
<i>Type</i>	165/220	75.0%	2/220	0.9%	15/220	6.8%	0.15
<i>Color</i>	155/203	76.4%	8/203	3.9%	20/203	9.9%	0.18
<i>Confirm</i>	407/455	89.4%	5/455	1.1%	9/455	2.0%	0.11

Table C.2 Close speech ASR error model trained on merged dialog corpus of WOz experiment and online-operation corpus

Average error angle	Standard deviation error angle	Recall Rate
21.1	4.54	52.2%

Table C.3 3D pointing gesture recognizer error model used for simulation retraining

Bibliography

- [1] Pietquin, O., **A Framework for Unsupervised Learning of Dialogue Strategies**, *PhD Thesis at Faculté Polytechnique de Mons*, 2004
- [2] Goddeau, D., Meng, H., Polifroni, J., Schreff, S., Susayapongchai, S., **A Form-Based Dialogue Manager for Spoken Language Applications**, *Proceedings of the International Conference on Spoken Language Processing*, 1996
- [3] Raux, A., Langner, B., Bohus, D., Black, A., Eskenazi, M., **Let's Go Public! Taking a Spoken Dialog System to the Real World**, *Proceedings of Interspeech*, 2005
- [4] Hipp, R., Smith, R., **A Demonstration of the Circuit Fix-it Shoppe**, *National Conference on Artificial Intelligence*, 1993
- [5] Litman, D., Sillimann, S., **ITSPOKE: An Intelligent Tutoring Spoken Dialogue System**, *Proceedings of the Human Language Technology Conference*, 2004
- [6] Matsusaka, Y., Fujie, S., Kobayashi, T., **Modeling of Conversational Strategy for the Robot Participating in the Group Conversation**, *Proceedings of Eurospeech*, 2001
- [7] Waibel, A., Lee, K.-F., **Readings in Speech Recognition**, *Morgan Kaufmann Publishers*, 1990
- [8] Huang, X., Acero, A., Hong, H.-W., **Spoken Language Processing: A Guide to Theory, Algorithm and System Development**, *Prentice Hall PTR; 1st edition*, 2001

-
- [9] Sutton, R.S., Barto, A. G., **Reinforcement Learning - An introduction**, MIT Press, 1998
- [10] Watkins, C.J., **Learning from Delayed Rewards**, PhD Thesis at Cambridge University, 1989
- [11] Levin, E., Pieraccini, R., Eckert, W., **Using Markov Decision Process for Learning Dialogue Strategies**, *Proceedings of the IEEE, Transactions on Speech and Audio Processing*, 1998
- [12] Levin, E., Pieraccini, R., Eckert, W., **A Stochastic Model of Human-Machine Interaction for Learning Dialog Strategies**, *Proceedings of the IEEE ICASSP*, 2000
- [13] Pietquin, O., Renals, S., **ASR System Modeling for Automatic Evaluation and Optimization of Dialogue Systems**, *Proceedings of IEEE ICASSP*, 2002
- [14] Singh, S., Kearns, M., Litman, D., Walker, M., **Reinforcement Learning for Spoken Dialogue Systems**, *Conference on Neural Information Processing Systems*, 1999
- [15] Walker, M., **An Application of Reinforcement Learning to Dialogue Strategy Selection in a Spoken Dialogue System for Email**, *Journal of Artificial Intelligence*, 2000
- [16] Scheffler, K., **Automatic Design of Spoken Dialogue Systems**, PhD Thesis, 2002
- [17] Lin, B.S., Lee, L.-S., **Computer-Aided Analysis and Design for Spoken Dialogue Systems Based on Quantitative Simulations**, *IEEE Transactions on Speech and Audio Processing*, 2001
- [18] Chung, G., **Developing a Flexible Spoken Dialogue System Using Simulation**, *Proceedings of ACL*, 2004
- [19] López-Cózar, R., De la Torre, A., Segura, J., Rubio, J., **Assessment of Dialogue Systems by Means of a New Simulation Technique**, *Speech Communication*, 2003
- [20] Georgila, K., Henderson, J., Lemon, O., **Learning User Simulations for Information State Update Dialogue Systems**, *Proceedings of Eurospeech*, 2005

-
- [21] Eckert, W., Levin, E., Pieraccini, R., **User Modeling for Spoken Dialogue System Evaluation**, *Proceedings of IEEE ASR Workshop*, 1997
- [22] Scheffer, K., Young, S., **Corpus-Based Dialogue Simulation for Automatic Strategy Learning and Evaluation**, *Proceedings of NAACL Workshop on Adaptation in Dialogue Systems*, 2001
- [23] Goddeau, D., Pineau, J., **Fast Reinforcement Learning of Dialog Strategies**, *Proceedings of IEEE ICASSP*, 2000
- [24] Scheffler, K., Young, S., **Automatic Learning of Dialogue Strategy Using Dialogue Simulation and Reinforcement Learning**, *Proceedings of the Human Language Technology Conference*, 2002
- [25] Schatzmann, J., Stuttle, M., Weilhammer, K., Young, S., **Effects of the User Model on Simulation-Based Learning of Dialogue Strategies**, *IEEE Automatic Speech Recognition and Understanding Workshop*, 2005
- [26] Schatzmann, J., Georgila, K., Young, S., **Quantitative Evaluation of User Simulation Techniques for Spoken Dialogue Systems**, *6th SIGdial Workshop on Discourse and Dialogue*, 2005
- [27] Hone, K.S., Baber, C., **Using a Simulation Method to Predict the Transaction Time Effects of Applying Alternative Levels of Constraint to User Utterances Within Speech Interactive Dialogues**, *ESCA Workshop on Spoken Dialogue Systems*, 1995
- [28] Pietquin, O., Beaufort, R., **Comparing ASR Modeling Methods for Spoken Dialogue Simulation and Optimal Strategy Learning**, *European Conference on Speech Communication and Technology*, 2005
- [29] Kelley, J., **An Iterative Design Methodology for User-Friendly Natural Language Office Information Applications**, *ACM Transactions on Office Information Systems*, 1984
- [30] Ammicht, E., Gorin, A., Alonso, T., **Knowledge Collection for Natural Language Spoken Dialog Systems**, *Proceedings of Eurospeech*, 1999
- [31] Williams, J.D., Young, S., **Using Wizard-of-Oz Simulations to Bootstrap Reinforcement-Learning-Based Dialog Management Systems**, *Proceedings of the 4th SigDial Workshop on Discourse and Dialogue*, 2003

- [32] Jönsson, A., Dahlbäck, N., **Talking to a Computer is not Like Talking to Your Best Friend**, *First Scandinavian Conference on Artificial Intelligence*, 1988
- [33] Fügen, C., Holzapfel, H., Waibel A., **Tight Coupling of Speech Recognition and Dialog Management - Dialog-Context Dependent Grammar Weighting for Speech Recognition**, *Proceedings of the International Conference on Spoken Language Processing*, 2004
- [34] Asfour, T., Ude, A., Berns, K., Dillmann, R., **Control of ARMAR for the Realization of Anthropomorphic Motion Patterns**, *The second IEEE-RAS International Conference on Humanoid Robots*, 2001
- [35] Finke, M., Geutner, P., Hild, H., Kemp, T., Ries, K. Westphal, M., **The Karlsruhe-VERBMOBIL Speech Recognition Engine**, *Proceedings of the ICASSP*, 1997
- [36] Soltau, H., Metze, F., Fügen, C., Waibel, A., **A One-Pass Decoder Based on Polymorphic Linguistic Context Assignment**, *Proceedings of the Automatic Speech Recognition and Understanding Workshop, ASRU*, 2001
- [37] Holzapfel, H., **Towards Development of Multilingual Spoken Dialogue Systems**, *Proceedings of the 2nd Language and Technology Conference*, 2005
- [38] Nickel, K., Stiefelhagen, R., **Pointing Gesture Recognition Based on 3D-Tracking of Face, Hands and Head Orientation**, *International Conference on Multimodal Interfaces*, 2003
- [39] Denecke, M., **Rapid Prototyping for Spoken Dialogue Systems**, *Proceedings of the 19th International Conference on Computational Linguistics*, 2002
- [40] Nickel, K., **Erkennung von Zeigegesten basierend auf 3D -Tracking von Kopf und Händen**, *Diploma thesis, University of Karlsruhe*, 2003
- [41] Holzapfel, H., Fügen, C., Denecke, M., Waibel, A., **Integrating Emotional Cues into a Framework for Dialogue Management**, *IEEE 4th International Conference on Multimodal Interfaces*, 2002
- [42] Azad, P., Asfour, T., Dillmann, R., **Combining Appearance-Based and Model-Based Methods for Real-Time Object Recognition and 6D-Localization**, *Proceedings of the International Conference on Intelligent Robots and Systems*, 2006