ulm university universität
uulm

User-centred Adaptive Spoken Dialogue Modelling

DISSERTATION

zur Erlangung des akademischen Grades eines

DOKTOR-INGENIEURS

(DR.-ING.)

der Fakultät für Ingenieurwissenschaften,
Informatik und Psychologie der Universität Ulm

von

Stefan Ultes

aus Schwäbisch Hall

Betreuer:	Prof. Dr. Dr.-Ing. Wolfgang Minker Prof. Dr. Ramón López-Cózar Delgado
Amtierende/r Dekan/in:	Prof. Dr. Tina Seufert

Ulm, 20.11.2015

Abstract

This dissertation investigates novel concepts and methods for the automatic recognition of dynamic user properties using statistical supervised learning and for the integration of these properties into the dialogue management process. The aim is to model the course of the dialogue adaptive to the user.

Current commercial spoken dialogue systems usually do not account for dynamic user properties like the user's satisfaction level. Even in state-of-the-art research systems, this type of adaptation is usually missing. However, if the system was aware of these properties, it would be able to have a better understanding of the current situation and thus to react more appropriately. Therefore, the goal of this work is to introduce this type of user-centred adaptation by separating the problem into two sub-problems: recognising the user state, i.e., the dynamic user properties, and integrating its estimation of the user state into the dialogue management process.

Before these individual sub-problems are approached, first, the necessary background, which is important for understanding the content of this thesis, is described containing relevant information about spoken dialogue systems and supervised machine learning. Following this description of the background, research of others which aims at solving similar research problems is described including a clear distinction of their work to ours.

For the first sub-problem of deriving the user state, we consider four different user states: the user satisfaction (US), the perceived coherence of the system reaction, the emotional state, and the intoxication level. Automatic recognition of these user states is based on supervised statistical learning. As US is a universal property, an emphasis is put on its automatic recognition with a focus on how temporal information may be used for this recognition process. Here, we propose three novel approaches on how to improve the US recognition performance: introducing an error correction module into the classification process, exploiting temporal learning algorithms by using a modified Markovian model, and optimising the feature set used as input to statistical classification models. While all of our proposed approaches result in a significant performance improvement, the best performance boost is achieved with an optimised feature set. With this feature set, we are able to improve the performance achieving a high correlation.

Research on the recognition of the remaining three user states have also resulted in significant contributions to the state-of-the-art. For the automatic recognition of the perceived coherence of the system reaction, we are to our knowledge the first to connect aspects of the interaction with coherence. Exploiting this relationship, the problem is modelled as a statistical classification task. In order to improve the performance of speech-based emotion recognition, we propose two

novel approaches which add information about the speaker to the recognition process. Here, we show that having speaker-dependent emotion recognition has the potential to improve the overall recognition accuracy. With a comparative study on intoxication recogntion, we compare the recongition performance of machines with humans showing that machines may outperform humans on this task.

For the second sub-problem of rendering the dialogue management adaptive to the user state, we propose three novel approaches. The first approach uses rules to select the next system action. These rules are based on the current user state. Here, we are able to outperform non-adaptive strategies for a bus information dialogue system in terms of task success / dialogue completion as well as dialogue length and user satisfaction. Our second approach introduces the user state into a POMDP-based dialogue manager by either extending the user state or utilising the user state for modelling the reward function. For the latter, the resulting dialogue policy achieves a better task success rate / dialogue completion rate than conventional reward functions. While these two approaches incorporate mechanisms which are common for dialogue management, our third approach proposes a two-stage model: in the first stage, any dialogue manager may be used to create an ordered list of possible system actions. In the second stage, the possible change of the user state induced by each system action is predicted. The final system action is then selected accordingly. In a simulated experiment, we are able to show that our proposed approach results in an significant improvement in dialogue coherence reducing the number of non-coherent system actions.

During the work on these two problems, we have created open-source implementations of a Conditioned Hidden Markov Model library as well as the POMDP-based dialogue manager. Both implementations have been made accessible to the public. Furthermore, we have created an annotated corpus for the recognition of user satisfaction as used in this thesis as well as for the recognition of the perceived coherence.

For my parents, my wife, and my son.

Acknowledgements

The work conducted in this dissertation would not have been possible without the help of many people. First and foremost, I would like to express my highest gratitude to my supervisor Prof. Wolfgang Minker. His constant support and confidence in my work have been most crucial for the successful realisation of this dissertation. He has always been highly supportive and thoughtful in his advice and guidance. Over time, he has become more a mentor and a friend than a professor.

I also would like to extend my highest gratitude to my reviewer Prof. Ramón López-Cózar Delgado (University of Granada, Spain) for his interest in my research work. He provided valuable advice and new insights during our visits to Granada. In this context, I would also like to thank Prof. Zoraida Callejas Carrión and Prof. David Griol Barres for their interest in my work and our constructive and fruitful collaboration.

A very special thanks goes out to Dr.-Ing. Alexander Schmitt. He provided me with direction and technical insight which I well needed especially in the early stages of my work. I have always been able to count on his support and I would not have been able to finish my work on the dissertation without his help and advice.

For successfully completing a doctorate project, it is very important to have a good environment to work in. This has foremost been created by my colleagues (and former colleagues) at the Institue of Communications Engineering at Ulm University, namely Florian Nothdurft, Helmut Lang, Tobias Heinroth, Maxim Sidorov, Louisa Pragst, Sergey Zablotskiy, Kseniya Zablotskaya, Roman Sergienko, Gregor Bertrand, and Tatiana Gasanova. Within this supportive atmosphere, I was always sure to find someone ready to listen. I would also like to thank my former Master and Bachelor students Juliana Miehle, Matthias Kraus, Hüseyin Dikme, María Jesús Platero Sánchez, Robert ElChabb, and Savina Koleva.

Thank you to Sungjin Lee and Maxine Eskenazi for providing access to their user simulator. Without their help, I would not have been able to conduct my experiments.

On a personal level, I would like to thank my colleague Sebastian Hentzelt for countless coffee breaks and discussions about life. Both provided the necessary distraction which greatly helped to regain focus.

Finally, but most importantly, I would like to thank my family. My parents have provided constant support and encouragement to pursue higher education. Without the grounding and the roots they provide I would have never dared to pursue a doctoral degree. This is also true for my wife and my son. Only with my wife's constant support, her practical views and her understanding, all of this has been possible.

Contents

List of Figures

List of Tables

Acronyms

ADL	average dialogue length
AD	action-dependent
AI	action-independent
API	application programming interface
ASR	automatic speech recognition
CHMM	conditioned hidden Markov model
DCR	dialogue completion rate
DM	dialogue management
DR	dialogue register
EC	error correction
HC	handcrafted
HIS	hidden information state
HMM	hidden Markov model
IP	interaction parameters
IQ	interaction quality
IS	information state
JSGF	java speech grammar format
MFCC	mel frequency cepstral coefficient
MLP	multi-layer perceptron
ML	machine learning
MR/R	match rate per rater
NLG	natural language generation
NLU	natural language understanding
OWL	web ontology language
PAD	pleasure arousal dominance
POMDP	partially observable markov decision process
RL	rule learner
RMSE	root mean squared error
SDA	system dialogue act
SDM	spoken dialogue management
SDO	spoken dialogue ontology
SDS	spoken dialogue system

SLU	spoken language understanding
SMO	sequential minimal optimisation
SVM	support vector machine
TSR	task success rate
TTS	text-to-speech
UAR	unweighted average recall
UDA	user dialogue act

1

Introduction

In today's high-tech world, computers and technical systems may be found everywhere to help humans in their daily routine, e.g., booking flights, getting information about bus schedules, or finding desired information in general. However, the ways of operating these devices and communicating with them are often quite artificial, i.e., a special communication language must be learned. Some of these communication languages are merged over time into our regular set of communication skills, e.g., swiping on the touch screen. Still, communicating with fellow humans instead of a technical device is usually much easier and consequently it would be desirable if the same communication means would be available for technical systems. Moreover, enabling human-machine communication to be natural is a prerequisite for providing easy access and allowing non-experts to use complex services. While natural communication skills comprise, among others, gestures, eye gaze, and facial expressions, speech is by far the most universal and the most flexible of those skills. Many times, using speech results in faster and more efficient communication. Furthermore, using speech also allows to communicate with a technical system even though the user is doing something else, e.g., driving a car or preparing food. For both, the user is able to focus on the primary task easily. (Also, using a touch screen device with fingers covered in flour is undesirable.)

For integrating the capabilities of speech communication into a technical device, the concept of spoken dialogue system (SDS) has been developed. These systems are designed to process the speech input, identify the relevant content which is passed to the application and generate system responses. While initial systems were quite restrictive, the performance of SDSs has improved over time, extending restricted initial application domains to more complex information retrieval and question answering applications (Raux et al., 2003; D'Mello et al., 2012; Metze et al., 2014), e-commerce systems (Tsai, 2005), surveys applications (Stent et al., 2006), recommendations systems (Chai et al., 2002), e-learning and tutoring systems (Kopp et al., 2012; Litman and Silliman, 2004), in-car systems (Hofmann et al., 2014; He et al., 2013), remote control of devices and robots in smart environments (Skantze et al., 2014; Minker et al., 2010), healthcare and ambient assisted living systems (Bickmore et al., 2010; Saz et al., 2009), or embodied dialogue systems and companions (Horchak et al., 2014; Qu et al., 2014).

1.1 Motivation

With the vision of enabling spoken human-machine interaction to be as efficient and natural as human-human interaction, recent work focuses on improving and augmenting the capabilities of spoken dialogue systems (SDSs) to increase the acceptance of speech interfaces. Here, the following two key aspects have been found to be of special importance:

1. **Unrestricted user behaviour**: The SDS must not put any restrictions (or at least as few as possible) on the style and content of the user behaviour, e.g., the dialogue domain or the turn taking model.
2. **User-centred modelling**: The SDS must account for user aspects of the interaction, e.g., the user state, in addition to its linguistic content.

Recent advances rendering the dialogue systems statistical (e.g., (Lemon and Pietquin, 2012; Thomson and Young, 2010; Young et al., 2010)) have already achieved major improvements towards reducing the restrictions put on the user behaviour. Still, there are restrictions on the style of the interaction mostly realising strict turn-taking behaviour where one system action is followed by one user action.

Solely enabling human-computer-interaction to be as flexible as possible, however, is not sufficient to achieve natural human-like communication. As stated in key aspect two, the dialogue system should also take account for additional user-specific aspects of the interaction, i.e., the user state, to further increase the naturalness of the interaction. However, although there is high potential, not much research has been conducted yet in the field of user-adaptive dialogue where the system is able to change its behaviour according to the user state. This change may be a change in the speaking style, in the word order and the word choice of the utterance, in the choice of content provided in the utterance, or in the strategy in general, e.g., when to provide help, how to prevent errors, etc.

For better illustration of the high potential, an example dialogue is shown in Figure 1.1 where somebody is calling a bus information system to get the info about a bus leaving from “West North Avenue”. Unfortunately, the system (S) does not recognise correctly what the user (U) is saying and acts on the assumption that the user said “Eighth Avenue”. To identify situations like that, the system employs a grounding strategy by restating the user’s input while asking for the destination (lines 5,7,9). The user realises that the system has not correctly understood the destination and corrects the system instead of providing information about the destination (lines 6,8,10). In line 10, though, the user is already unsatisfied with the system (and probably also a bit angry) which changes the speaking style. These changes, though, result in the system not understanding the word “no” and the system understands that the user provided input for the destination. Thus, the system regards “West North Avenue” as confirmed origin, “Eighth Avenue” as unconfirmed destination and continues with asking for the departure time. Here, a system which was more user aware would have had the option to change the error prevention strategy.

The same dialogue in an adaptive version is shown in Figure 1.2. Here, the system tracks the user state and recognises that the user is frustrated, angry, and not satisfied with the interaction. Hence, it adapts the dialogue by changing the error preventing strategy: instead of restating the understood user input and asking for information at the same time, a closed question is posed asking the user to confirm the understood information explicitly (line 9). As this restricts the user input to only be affirmation or rejection (yes or no), the system is much less prone to errors

		Utterance	
(1)	S:	Welcome to the bus information system. What can I do for you?	
(2)	U:	54c Schedule.	☺
(3)	S:	The 54C. Where do you want to leave from?	
(4)	U:	West North Avenue.	☺
(5)	S:	Leaving from EIGHTH Avenue. Where do you want to go to?	
(6)	U:	No, leaving from West North Avenue.	☺
(7)	S:	Leaving from EIGHTH Avenue. Where do you want to go to?	
(8)	U:	No, leaving from West North Avenue.	😐
(9)	S:	Leaving from EIGHTH Avenue. Where do you want to go to?	
(10)	U:	[No.] West North Avenue.	😐
(11)	S:	Going to West North Avenue. When do you want to travel?	
(12)	U:	I want to leave from West North Avenue.	☹
(13)	S:	Sorry, I did not get that. When do you want to travel?	
(14)	U:	I need the next bus.	☹
(15)	S:	The next bus from eighth avenue to west north avenue leaves as 12.06 pm.	

Fig. 1.1: A non-adaptive dialogue example with implicit confirmation prompts. As is indicated by the emoticons, the user is unhappy and unsatisfied in the end.

in the recognition of the user's input. Hence, the user is only able to respond with a "no" (line 10) allowing the system to ask for the origin again. This dialogue is, without question, a simple example. This concept of adaptation, though, may easily be transferred to more complex dialogue situations (and user states).

		Utterance	
(1)	S:	Welcome to the bus information system. What can I do for you?	
(2)	U:	54c Schedule.	☺
(3)	S:	The 54C. Where do you want to leave from?	
(4)	U:	West North Avenue.	☺
(5)	S:	Leaving from EIGHTH Avenue. Where do you want to go to?	
(6)	U:	No, leaving from West North Avenue.	☺
(7)	S:	Leaving from EIGHTH Avenue. Where do you want to go to?	
(8)	U:	No, leaving from West North Avenue.	😐
(9)	S:	Leaving from EIGHTH Avenue. Is this correct?	
(10)	U:	No.	😐
(11)	S:	Where do you want to leave from?	
(12)	U:	West North Avenue.	😐
		⋮	

Fig. 1.2: An adaptive dialogue example starting with implicit confirmation prompts (1-7) and the switching to explicit confirmation (9). In contrast to Figure 1.1, the user does not get completely unsatisfied with the interaction due to the strategy change.

To equip a dialogue system with the ability to react differently according to the user state, the system needs to be enhanced in two ways: means of identifying the user state and means of taking

into account the user state must be integrated into the system. Consequently, in this thesis, we will conduct research in both fields, automatic *user state recognition* and *user-adaptive dialogue modelling*, where approaches on dialogue modelling will be built upon knowledge gained form recognising the user state. For user state recognition, we have selected the dynamic user states *user satisfaction*, *perceived coherence*, *emotions* and even *intoxication*. The former two states are very general and may be applied to almost all types of dialogue. Furthermore, to enable more natural interaction, it is crucial for the dialogue system to be able to identify more vague states like the user's emotion or even their intoxication level. If the system was aware of these states, it would be able to tailor its reactions accordingly.

For all states, we will provide novel work on recognising these user states. Furthermore, we will propose methods of using the user state information to influence the decision of selecting the next system action. The contributions we intend to provide with this thesis in both fields are described in the following.

1.2 Thesis Contributions

For advancing towards the goal of rendering the human-machine interaction natural by means of user-centred adaptive dialogue, the overall problem is divided into the fields of user state recognition and user-adaptive dialogue modelling.

In the field of **user state recognition**, we will provide novel approaches which aim at improving the recognition performance for all four user states under consideration. For recognising the user satisfaction, we will investigate if there is a strong link between the user satisfaction and temporal context, i.e., previous interaction steps and the user's satisfaction at those points, and present work on exploiting this relation to improve the overall recognition performance.

Furthermore, to recognise the perceived coherence of the system behaviour, we will examine if the coherence may be related to events of the interaction by statistical modelling. Here, we will employ a classification approach taking into account the system action, the current dialogue, and parameters reflecting events of the interaction. For speech-based emotion recognition, we will conduct research on whether information about the speaker may be included into the recognition process with the perspective of improving the recognition performance.

Within the field of **user-adaptive dialogue modelling**, we will develop and evaluate approaches on introducing user-state adaptivity into the process of selecting the next system action. Here, we will provide evidence for rule-based and statistical systems that this is worthwhile and will lead to improvement of the overall dialogue performance given the user state as well as objective performance measures. More precisely, we will investigate if there is a correlation between the user satisfaction and task success or average dialogue length. Moreover, we are interested in the usability and applicability of user satisfaction for dialogue systems based on reinforcement learning where the system learns what good actions are by getting a reward. Here, we are especially interested if the user satisfaction may be used to model that reward. Within several evaluations of adaptive dialogue, we will show that both rule-based adaptation as well as using the user state for modelling the reward result in an significant improvement of the dialogue performance.

Finally, we will investigate how the perceived coherence may be used to **improve the dialogue**. Here, we will present a novel approach on selecting the next system action by taking into

account the expected change in perceived coherence on the user. By modelling this as a two-stage process, the coherence-based modifications may be added to virtually any dialogue system. The validity of this approach will be evaluated within a corpus-based experiment.

1.3 Outline

The structure of this thesis is as follows: for readers who are not familiar with the general concepts of dialogue systems, dialogue management, or machine learning, the relevant background is described in Chapter 2. In Chapter 3, we place this work within the work by others. There, we will both present related work on user state recognition for the four user states *user satisfaction*, *perceived coherence*, *emotions*, and *intoxication* and present related work on adaptive user modelling taking into account the user state. The first part of our main contribution comprising approaches on recognising the user state will be described and evaluated in Chapter 4. For each of the four investigated user states, our proposed approaches together with their appertaining experiments will be presented in their respective own section. The second part of our main contribution—approaches on user-adaptive dialogue modelling—will be thoroughly described and evaluated in Chapter 5. There, we will present our approaches on adapting the course of the dialogue to the recognised user state and evaluate their viability for adapting basic dialogue strategy concepts. Furthermore, we will present work on using the perceived coherence for selecting the next system dialogue act. Finally, this thesis will conclude in Chapter 6 with a discussion of all results along with a prospect on related work in the field of statistical modelling of user-centred adaptive spoken dialogue.

2

Relevant Background

For understanding the presented work on user-adaptive speech interfaces which is described in this thesis, we first have to establish a common background and explain all relevant basics. This includes, of course, a detailed description of how a Spoken Dialogue System (SDS) and its components work. In fact, the operation method of an SDS is important throughout the thesis even for identifying the user state, where information about the interaction is used which is directly derived from specific modules of the SDS. Along with a general description of the SDS, a closer look is taken at the dialogue manager which relies at the core of a human-machine speech interface (Minker et al., 2009). While we briefly touch important approaches to dialogue management, two are described in more detail as they are used for adaptive dialogue modelling: the Information State (IS) approach by Larsson and Traum (2000) and the Hidden Information State (HIS) approach by Young et al. (2007). These two have been chosen as they are based on the same idea. In fact, the latter is an extension of the former one providing probabilistic dialogue management functionality. We will use these two approaches to adapt the course of the ongoing dialogue based on a predefined dialogue strategy taking into account the user state. Common dialogue strategies may incorporate fundamental strategy concepts. The two most important concepts, which are also relevant in this thesis, are grounding and the dialogue initiative. In order to understand how well a dialogue system—more precisely the employed strategy—is working, the system should be evaluated. Here, fundamentally different concepts are used: objective measures like the length of the dialogue versus collecting the user's opinion. Candidates from both concepts are used within this thesis and are also discussed in detail.

For providing a user-centred adaptive dialogue flow, the user state needs to be known. Usually, it is impossible to ask the users about their state (e.g., this would disturb the interaction). Instead, the user state may be estimated. A common approach to solve estimating tasks like the one at hand is to formulate the problem as a classification or supervised machine learning task. There, a discrete set of classes is estimated based on a set of features. Hence, the approaches used within this thesis are described in detail. They are all based on the general concept of statistical machine learning. These machine learning algorithms will be applied to several user state estimation approaches and evaluated. These user states are described in Chapter 4. For evaluating the estimation performances, four different metrics are used presented in this chapter. The metrics provide different characteristics and thus offer insights into the performance of the machine learning approaches from different angles.

Before we present details about statistical machine learning, though, the general concept of Spoken Dialogue Systems will be described in the following.

2.1 Spoken Dialogue Systems

Spoken Dialogue Systems are multidisciplinary systems—even from a technical point of view—and have a long history. While there have been several non-speech dialogue systems around, the first system dates back to the early 1990s with the VOYAGER system (Zue et al., 1991). Enabled by the advent of automatic speech recognition and language processing capabilities, the VOYAGER system provides information for urban exploration in Cambridge, Massachusetts, USA in a very simplistic and restricted manner. Since then, the capabilities of spoken dialogue systems have increased considerably. State-of-the-art systems are able to handle uncertainty efficiently using probabilistic models or even allow for incremental processing. Regardless, even today's systems are still based on the same general architecture.

2.1.1 General Architecture of Spoken Dialogue Systems

To provide speech interaction between an application and the user—having a speech audio signal as input and output—the SDS solves several sub-tasks. The input audio signal is pre-processed and transferred into written words. These words are then semantically interpreted and put in context to the previous interaction in order for the system to decide how to react. This reaction may include both the communication with the user and the communication with the back-end application. For generating a system response, the system generates an utterance (words or sentences of what to say) which is then in turn transformed into a speech signal. As all problems may be tackled separately, a modular technical system is created—employing the divide-and-conquer paradigm—having each module independent and specialised on the respective sub-task.

The modular architecture of an SDS is presented in Figure 2.1 and contains the following modules:

Speech Recognition	Based on pre-processed audio speech input, the speech signal is transformed into the most likely sequence of words.
Language Understanding	Completely independent from the audio signal, the sequence of words is semantically interpreted and annotated with the most likely meaning.
Dialogue Management	The core component of the SDS interprets the semantically annotated input based on the previous interaction. The dialogue manager then decides with respect to the task at hand which system action to execute.
Text Generation	The abstract system action is transformed into a sequence of words: the actual system utterance.
Speech Synthesis	To generate the audio response and provide it to the user, the speech is synthesised by transforming the words into an audio signal.

While speech recognition and language understanding are on the input side of the SDS, language generation and speech synthesis form the output of the system. In the following, we will give a more detailed description of each component based on the execution order.

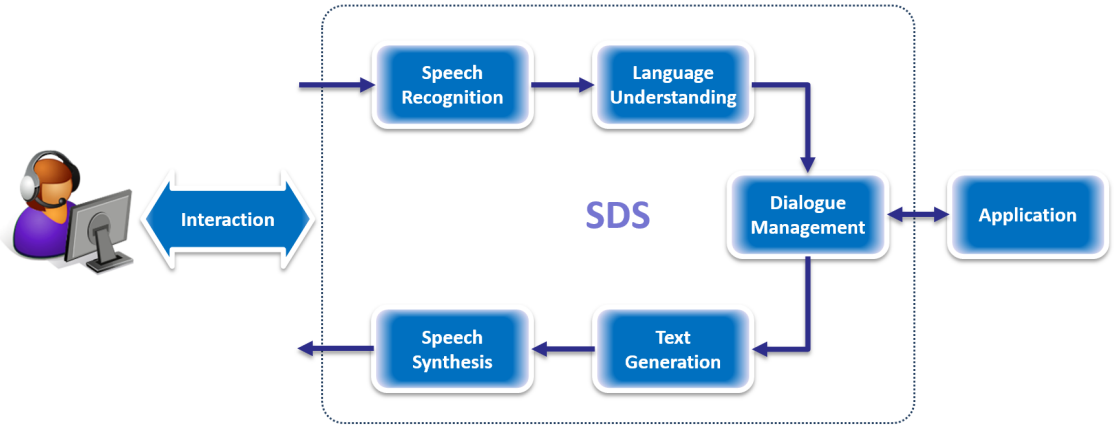


Fig. 2.1: The general architecture of an SDS as in (Lamel et al., 2000): speech recognition and language understanding on the input side, language generation and speech synthesis on the output side, and in between the dialogue manager handling the complete interaction with the user and the application.

Automatic Speech Recognition

The goal of Automatic Speech Recognition (ASR) is to transform spoken words into written form. More precisely, based on a sequence of feature vectors x, the most likely word sequence $\hat{w}$ is estimated. The ASR pipeline contains an initial pre-processing step including the feature extraction generating the vectors x followed by the actual speech recognition. This overall pipeline depicted in Figure 2.2 will be explained module by module in the following.

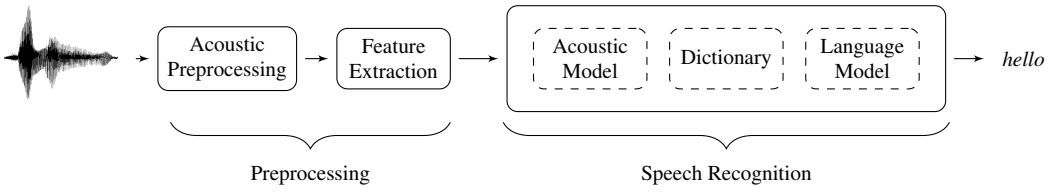


Fig. 2.2: The pipeline for automatic speech recognition[1]: after preprocessing, the feature vector is used for speech recognition to map the input audio signal to a sequence of words (here: “hello”).

Starting from the digitalised speech signal, acoustic pre-processing methods may be applied, e.g., a Wiener Filter (Wiener, 1949) to preform noise reduction. Based on this pre-processed signal, the features needed for speech recognition are extracted. Feature extraction is necessary to only preserve the information of the speech signal relevant for speech recognition and discard the rest. Here, the most important information lies within the dominant frequencies in the speech signal (vocals, e.g., are identified by the first and second formant, a harmonic of the fundamental

[1] Wave form in Figures 2.2, 2.4, 3.2, 4.26, 4.27 from www.freesound.org under CreativeCommons license 3.0: `http://creativecommons.org/licenses/by/3.0/legalcode`

frequency). Hence, a frequency analysis is performed using Fourier transformation calculating the spectrum of the signal. As the human ear is receptive for certain frequencies bands, a Mel filter is applied condensing the frequencies of the spectrum to 13 remaining frequency features. These features are further projected into the cepstrum space by applying an inverse cosine transformation. The resulting features are the Mel Frequency Cepstral Coefficients (MFCCs), which are then used for speech recognition[2].

State-of-the-art automatic speech recognition systems use statistical models. Based on the vectors x containing MFCCS, the most likely word sequence $\hat{w}$ is estimated by calculating the probabilities $p(w|x)$ for all possible word sequences and taking the word sequence with the maximum probability

$$\hat{w} = \arg\max_{w} p(w|x) \; . \tag{2.1}$$

As $p(w|x)$ is hard to be estimated directly, the Bayes rule is applied:

$$\arg\max_{w} p(w|x) = \arg\max_{w} \frac{p(x|w)P(w)}{p(x)} \; . \tag{2.2}$$

As $p(x)$ is independent of the argument w to maximise over, this equation can be further reduced to

$$\arg\max_{w} p(w|x) = \arg\max_{w} p(x|w)P(w) \; . \tag{2.3}$$

Using this simplified equation, only the probability models $p(x|w)$ and $P(w)$ have to be estimated. Here, the former is known as the acoustic model representing the actual mapping from the audio signal to the word sequence under consideration of a dictionary. The latter is the language model which allows to incorporate probabilities for certain word sequences applying restrictions induced by grammar and syntactic rules.

Acoustic Model The acoustic model calculates the probability $p(w|x)$ by calculating the most probable sequence of phonemes[3] which best represent the input vectors. Using a dictionary, the phoneme sequence is then mapped to actual words.
Most state-of-the-art commercial speech recognisers use Hidden Markov Models (HMM) (Rabiner, 1989). As we will use HMMs to estimate the user state within this thesis, those are not described here. Instead, a detailed description may be found in Section 2.2.2. To understand how HMM are used for acoustic modelling, it is sufficient to know that the goal is to identify the most probable state sequences which represent the audio signal best. As each state is then tied to a phoneme, possible sequences of phonemes may be identified. The most likely sequence is then selected with respect to the entries in the dictionary: only phoneme sequences which have an entry in the dictionary are recognised.

Language Model To further add language specific information the word sequence, the language model is used. Conventional systems may use a context-free grammar to define which word sequences (or sentences) may be recognised simply by setting a probability of 1.0 for sequences modelled by the grammar and 0.0 for all others. However, spoken language is often not very grammatical so it is a nearly

[2] The MFCCs may also be used for estimating emotions or intoxication. There, however, they are only a small part of all used input variables

[3] Phonemes are the basic unit of speech.

impossible task to generate a grammar modelling each possible valid user input. Hence, many advanced ASR systems nowadays use statistical models. So called *n*-grams model the probability of a word sequence containing *n* words. Hence, a tri-gram ($n = 3$) models the probability $p(w_n|w_{n-1}, w_{n-2})$ of a sequence of three words. Such a model is based on the Markov assumption which states that the current state only depends on a finite number of previous states (for $n = 3$ on two previous states, i.e., words). This simplification is reasonable (and even necessary) as modelling each possible word sequence would result in a model which would not generalise very well. To create the *n*-grams for the language models (usually, bi- or tri-grams are used), training data may be used to easily compute the probabilities based on the respective occurrences in the training data.

Now having seen how the individual modules work, it may still be unclear how they collaborate to generate the final hypothesis. To create the search space, for each entry in the dictionary, an HMM with the respective phoneme states is created. Furthermore, the HMMs are connected according to the *n*-grams. Evidently, this will result in many paths and a huge space where it is very time consuming to find the most probable word sequence. To handle this complexity, advanced search methods like beam search are used. Here, only the most promising paths are regarded and paths with a probability below a certain threshold are pruned. Based on this, a confidence is calculated representing the certainty of the algorithm that the estimated word sequence $\hat{w}$ is correct.

However, the estimation is not always correct and the correct word sequence is not always the one with the highest confidence but, e.g., the estimated with the third highest confidence. To allow following modules to take account for this, most state-of-the-art systems allow for creating *n*-best-lists as result. An *n*-best-list contains the *n* estimated word sequences with the highest confidence. This list containing the conversion from spoken language to text is then used within the Language Understanding module to extract meaning.

Language Understanding

Having a text representation of the spoken input, it is still unclear what the actual meaning is. This task of extracting the meaning is fulfilled by the natural or spoken language understanding (NLU/SLU) module by doing a linguistic analysis to extract the semantics of the text representation by using grammatical relations, rule-based semantic grammars, template matching, or statistically driven techniques (Allen, 1995; Jurafsky and Martin, 2008). One approach often used in commercial systems is to have a context-free grammar shared with the language model of the ASR. Here, each rule is associated with one or more tags which allow the grammar designer to add semantic information to each rule. The example in Figure 2.3 in the Java Speech Grammar Format[4] (JSGF) shows a set of grammar rules which allow the user to provide information about a destination. Based on the tags, the rules also provide the semantic interpretation, e.g., with `orig="value"`.

In general, McTear (2004) identifies syntactic and semantic approaches for the process of language understanding or semantic analysis. Syntactic approaches aim at gaining information

[4] `http://www.w3.org/TR/jsgf/`

```
public <root> = I want leave from <origin>;
     <origin> = West North Avenue {orig="west north avenue"}
              | Eighth Avenue {orig="eighth avenue"};
```

Fig. 2.3: An example for a JSGF grammar with added tags for semantic interpretation.

about the meaning by analysing the syntactic structure. By identifying subsequently phrases, e.g., verb or noun phrases, using context-free grammars, a syntactical tree is created allowing for associating content words with their semantic function. However, this association is not trivial. Hence, recent approaches apply data-driven methods and machine learning algorithms like hidden understanding models (Miller et al., 1994; Minker et al., 1999). Semantic approaches also use grammars. However, the grammar rules aim not at identifying the syntactic role of the words or phrases. Instead, they model the semantic structure directly. A simple instance of those grammars has already been described in the beginning of this section.

The extracted meaning is then directly fed into the dialogue manager for further processing.

Dialogue Management

Having a semantic interpretation, the system needs to provide access to the back-end application and to decide about the next system action. While this is a simple task for non-complex question-answer systems where one system reaction follows one user input without keeping context information, this is more complex for real task-oriented dialogues: the system needs to know about the task to solve and about the information which is needed for this. Being able to keep context information over multiple queries, the system is able to easily collect missing information without the need for the user to restate anything. To provide this type of functionality, a dedicated module is needed. This dialogue manager processes the semantic input of the user and interprets it with respect to the context of the current dialogue. Usually, keeping this context information over a sequence of turns is achieved with a dialogue state capturing relevant information. Based on this dialogue state—updated with on the semantics of the user input—and the task to solve, the system decides about the next system action. For this decision, however, different strategies may be executed, which may all follow the same goal in different ways. This is elaborated in Section 2.1.3. The selected system action is then provided to the language generation module.

To manage the interaction, several approaches have been proposed. The simplest approach is a finite state based dialogue manager. The course of the dialogue is pre-set and very inflexible. This also restricts the user to a very rigid communication style. More flexible approaches include a slot-filling approach. In a task-oriented dialogue, the task may be decomposed into several slots, i.e., grouped information belonging to one semantic concept. For our example of a bus schedule dialogue (Chapter 1), the slots may be departure place, destination, departure time, and bus route. Now the user is free to address any slot. Dependent on the user input, the system may then only address slots with missing information. Of course, there are more advanced dialogue management methods. The approaches relevant for this thesis will be presented in detail in Section 2.1.2.

Natural Language Generation

The output of the dialogue manager output is an abstract system action, e.g., `welcome()` or `request(slot)`, possibly with an associated parameter as in the latter example. However, for giving feedback to the user in a natural way using speech, this abstract representation first needs to be transferred into a sequence of words. To generate this text is the task of the natural language generation module.

A simplistic approach but well functioning for most systems currently deployed is template filling. A template of the text for each system action is stored, e.g., *Welcome to the bus schedule system. How may I help you?* for the system action `welcome()`. These templates may also contain placeholders for content specified by the parameters of the system action. For the system action `request(slot)`, the corresponding template might be *What is your <slot>?* (*What is your destination?* for `request(destination)`). Of course, this approach highly depends on the system designer and does not offer a lot of variety which might bore the user at some point.

For text-based systems, i.e., simply showing the system response on a display, the dialogue system's turn would be finished. However, having speech output, the response need to be transformed into spoken words which will be described in the following.

Speech Synthesis

To have a spoken language response to the user, the first systems used recorded utterances: each abstract system action from the DM triggered the playback of a audio file. For this, of course, no NLG component was used. As using audio playback is a very inflexible approach, modern systems generate a speech signal out of text. These text-to-speech (TTS) systems are usually designed as a two-part problem as depicted in Figure 2.4: first, the parts of the text (words etc.) are translated to a sequence of phonemes. This grapheme to phoneme (g2p) mapping may use a dictionary in the simplest case. However, words which are not contained in the dictionary may not be processed. Hence, modern systems use machine learning approaches, e.g., decision trees (Andersen et al., 1996), to automatically generate the phoneme sequence out of words.

The second part of the TTS is the actual synthesis based on a sequence of phonemes. One approach is to use a database of audio recordings for each phoneme. To generate the speech signal, the respective audio phoneme snippets are selected and simply concatenated. To generate a more natural speech flow, the hard transitions are smoothed. Other approaches generate the speech signal for each phoneme artificially (in a reverse manner compared to ASR). Using both approaches simply as described would result in a very monotone voice. Hence, prosody is added to generate more natural and interesting voices. As speech synthesis does not play a major role within this thesis besides its plain usage, it is not described in more detail.

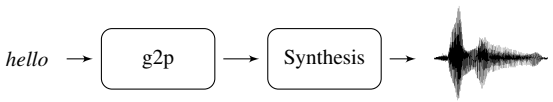


Fig. 2.4: The pipeline for text-to-speech modules: first, the text sequence must be translated into a sequence of phonemes (g2p). Based on these phonemes, the speech is synthesised.

Up to his point, the basic functionality of all modules of the SDS has been described. As one main aspect of this thesis lies in the area of dialogue management, a more detailed look is taken at two approaches to dialogue management.

2.1.2 Relevant Approaches to Spoken Dialogue Management

As described above, the dialogue manager resides at the core of an SDS being responsible for maintaining or tracking the current state of the dialogue, selecting the next system action based on the current state and the task at hand by following a strategy or policy, and providing access to the back-end application. Starting from simple state-based approaches, more advanced methods have been developed, e.g., frame-based dialogue management (Seneff and Polifroni, 2000) or agent-based dialogue management (Bohus and Rudnicky, 2003). All management approaches have their strengths and weaknesses depending on the task to solve.

In this thesis, we are interested in adapting the course of the dialogue to the user state. More specifically, we want to compare rule-based adaptation to more complex techniques where the optimal adaptive strategy is automatically learned in a probabilistic system. For this, we utilise two dialogue management approaches which are based on the same idea. The only difference is in the way they maintain the dialogue state and the resulting options of deriving the next system action. By that, we will be able to create one unified dialogue management software which uses the same dialogue description for both. The shared idea of the used approaches is called *Information State* and will be elaborated in the following.

The Information State Approach to Dialogue Management

The term Information State (IS) dates back to 1996 as an extension to the finite-state based dialogue modelling theory common at that time. The IS is defined as all information which is needed to proceed with the dialogue (Ginzburg, 1996). As this allows for dialogues with no predefined execution sequence, IS-based systems are capable of generating dialogues which are more flexible than finite-state based systems.

Larsson and Traum (2000) have further specified the IS theory in order to provide a common theory actual dialogue management approaches may be mapped onto, including finite-state based systems[5]. For IS-based systems, all relevant information within a dialogue may be identified as well as how this information is updated and how the updated is regulated. Larsson and Traum (2000) emphasise the update of the IS triggered by abstract system and user actions as key aspects. Even more so, the IS theory defined by Larsson and Traum (2000) consist of five main pillars which are explained in the following.

Informational components The informational components describe all components relevant for the dialogue which carry information. Here, relevant information is defined as being necessary in order for the system to create the desired behaviour. This may be information about the context or information about the interaction itself. Examples for this are the user's internal state (e.g., user's belief, intention, or goal) or information related to

[5] Matching finite-state based systems to IS is trivial while it may be impossible to find an IS formulation for a given finite-state based dialogue (Larsson and Traum, 2000).

	the structure of the dialogue (e.g., grounding status). Furthermore, the components of information state may also be grouped into *dynamic* and *static* information. Static information does not change during the course of the dialogue and may thus be handled differently than dynamic information.
Formal representation	To model the informational components, a formal representation is used. There, the data structure for the components may be defined, e.g., OWL ontologies (Antoniou and van Harmelen, 2004) or typed feature structures (Carpenter, 1992). Furthermore, the accessibility of the information may be defined, e.g., a FIFO queue, a LIFO stack, or a random access structure. Both, the data structure and the accessibility should match the implemented theory of the dialogue manager.
Dialogue moves	As in natural language spoken dialogue an almost infinite number of different utterances may possibly refer to the same concept, dialogue moves are meant to form an abstraction layer. More precisely, the dialogue moves modelling abstract user actions trigger the state updates. Hence, the number of dialogue moves is limited by the number of different updates necessary to model the problem as well as by natural language understanding capabilities. In addition, dialogue moves may also model abstract system actions which can be executed.
Update rules	Updating the information state is subject to a set of predefined rules. Conditioned on aspects of the current information state, the rules are executed and change the information state as the dialogue progresses.
Update strategy	Having a set of update rules, the update strategy defines which rules are applied to update the IS if the conditions of more than one rule are met, e.g., always selecting the first rule. Further detailed definitions are possible, e.g., having different strategies for processing user input and selecting the system move.

In addition to defining these five components of the IS theory, Larsson and Traum (2000) also introduced their realisation of the theory. They describe the tool kit TrindiKit providing a general installation framework for IS-based dialogue managers. More precisely, they provide the general architecture so that only the five IS components need to be realised. Hence, the framework consists of the Information State itself, a dialogue move engine containing the update mechanisms of the IS as well as the selection mechanism for the next system move, interfaces to other relevant modules of the dialogue system (e.g., language understanding or language generation), and a control module linking all components together. The TrindiKit has been used to implement several actual dialogue managers, e.g., (Bohlin et al., 1999; Matheson et al., 2000), showing the general viability of the approach.

However, the TrindiKit is very heavy-weighted and inflexible. Due to its purpose of providing a general framework applicable to all kinds of dialogue managers, TrindiKit needs to introduce a lot of computational and implementational overhead. Furthermore, it needs an extensive rule base to model the dialogue flow defining update and selection mechanisms. Here, the rule base has to be kept in a coherent state. For complex dialogues, achieving this is a very complex task. Thus, TrindiKit does not provide easy extensibility.

As an extension to the information states, hidden information states have been introduced which will be described in the following.

The Hidden Information State Approach to Dialogue Management

In general, Spoken Dialogue Systems have to deal with uncertainty, e.g., regarding the actual user action. This uncertainty is accumulated over multiple stages of the dialogue cycle. To handle this uncertainty, the concept of the Information State approach has been extended resulting in the Hidden Information State (HIS) approach proposed by Young et al. (2007) which is based on the formal concept of Partially Observable Markov Decision Processes (POMDPs). In the HIS, the uncertainty is accounted of by handling several information states with associated probability. Thus, the "real" state is hidden (comparable to HMMs, see Sec. 2.2.2).

Formally, a POMDP (Kaelbling et al., 1998) is defined as the 6-tuple (S,O,A,T,Z,b_0). It consists of a set S of state variables s, a set A of system actions a, and a set O of all possible observations o of the system. Furthermore, transition probabilities $P(s'|s,a) \in T$ and observation probabilities $P(o'|s',a) \in Z$ model the uncertainty. As the state of the underlying process cannot be determined exactly, a probability distribution over all possible states, called the *belief state* $b(s)$ with b_0 as the initial state, is used instead. There are two tasks associated: to update $b(s)$ based on a new observation and to identify the best action to take with respect to the current belief state. The latter is done by finding the optimal policy $\pi^*(b)$ which maximises an overall reward (to each action a and state s, a reward $r(s,a)$ is associated). This is usually done using reinforcement learning techniques.

Williams and Young (2007) introduced this concept into the world of spoken dialogue systems as POMDPs offer a unified mathematical model to handle uncertainty. Three features of POMDPs applied to SDSs are of particular interest:

- input result lists with confidence scores
- parallel state hypotheses
- automated action selection.

By using multiple hypotheses of what was said by the user, the generated *result list* in combination with associated *confidence scores* constitute an n-best-list containing the n best hypotheses of what was said by the user (instead of just the most likely one). These are incorporated into *parallel state hypotheses* representing multiple possible dialogue states at the same time. By this, misunderstandings of the system do not automatically cause other information to be lost. *Automated action selection* based on a probability distribution over all state hypotheses is executed. An optimal policy is trained based on example dialogues to automatically find the best system action for each dialogue state.

To cast a dialogue system as a POMDP, first, the state s is decomposed into (u,g,h) representing user action u, user goal g, and dialogue history h as proposed by Williams and Young (2007), who also introduce reasonable independence assumptions. Hence, the general POMDP equation for updating the belief $b(s)$ of state s

$$b'(s') = p(o'|s') \cdot \sum_{s} P(s'|s,a) \cdot b(s) \,, \tag{2.4}$$

where o' is the current observation and a the last system action, is transformed to

$$b'(u',g',h') = k \cdot P(o'|u') \cdot P(u'|g',a) \cdot \sum_{g} P(g'|g,a) \cdot \sum_{h} P(h'|u',g',h,a) \cdot \sum_{u} b(u,g,h) \; . \tag{2.5}$$

The observation probability $P(o|u)$ is estimated by $P(o|u) \approx P(u|o)$. Moreover, $P(u|o)$ is directly estimated by taking the confidence scores of the n-best list entries provided as result from the ASR and NLU modules.

However, casting an SDS as a POMDP yields the problem that computing a probability distribution over all dialogue states is intractable. Hence, Young et al. (2007) combined the ideas of POMDPs in SDSs and the Information State approach resulting in the Hidden Information State (HIS) approach to dialogue management. Instead of modelling the complete state space, only partitions are used. Each partition may then be regarded as on representation of an individual information state.

More precisely, assuming a slot-filling dialogue, the user goal space is partitioned into equivalence classes, or partitions p, according to the possible values a slot can take. Introducing further simplification, for each slot, a partition represents either all possible slot values, one specific slot value, or the partition may exclude a set of values for the respective slot.

When new user input arrives, the belief state is updated in two phases. First, the partitions are split according to the new user input. This includes distributing the probability mass of the originating partition to the resulting partitions. In the second phase, the belief is updated according to equation

$$b'(u',p',h') = k \cdot P(o'|u') \cdot P(u'|p',a) \cdot \sum_{h} P(h'|u',p',h,a) \cdot \sum_{u} P(p'|p) \cdot b(u,p,h) \; . \tag{2.6}$$

Here, $P(p'|p)$ denotes the probability of partition p' originating from partition p or, in other words, the fraction of probability mass which is transferred from p to p' if p is split into p' and $p - p'$.

According to Williams (2010b), the splitting probability

$$P(p'|p) = \frac{b_0(p')}{b_0(p)} \tag{2.7}$$

is computed as the ratio of the prior probability of the new partition $b_0(p')$ to the prior probability of the originating partition $b_0(p)$. Williams (2010b) compute the prior probability by counting all possible user goals, i.e., combinations of slot values, the partition represents and dividing this number by the total number of distinct user goals.

For better illustration of the partitioning approach, an example in the bus schedule information domain (the domain from the dialogue examples in Chapter 1) is shown in Figure 2.5 with a reduced number of slots and slot values. Initially, there is only one partition containing all values for the two slots `origin` and `destination`. If new user input arrives, the partition is split according to the slot the user input belongs to. In this example, the n-best list entries target the slot `destination` and comprise two entries. Therefore, the root partition is split and two new partitions are created, each one containing one of the two values in the slot `destination`. The range of slot-values the original partition is representing has been reduced to exclude the two destinations provided by the user. Following that, the new belief values are determined. In order to select the next system action, the summary belief is computed. Based on this, the policy is applied and the resulting system action is refined and executed. For computing the prior probabilities for

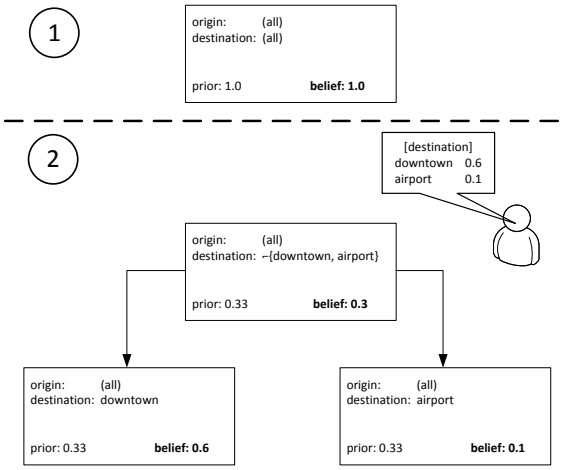


Fig. 2.5: An example of partition splitting with three possible destinations originally published in (Ultes et al., 2014a): "airport", "downtown", and "train station". First, there is only one partition subsuming all values for the two slots. After splitting the partition on the user input, two new partitions are created each representing all goals containing "airport" or "downtown", respectively, as destination, while the original partition excludes both values.

this example with three possible values for origin and destination respectively, the total number of distinct goals is 9. Therefore, by limiting the destination in all three partitions of step 2 to one single value, each remaining partition represents 3 remaining goals. Hence, the prior probability of all partitions is 0.33.

For policy execution, the belief state (the partition tree) is transformed into a summary belief space only containing information about the two most probable partitions. Reducing the dimensions of the state space is necessary to render the optimisation process more tractable. Based on this, a summary system action is selected according to the trained policy. The summary system action then has to be refined using heuristics. The resulting action is then executed by the system.

In order to find an optimal policy, reinforcement learning is applied based on the value function $V^{\pi}(b_t)$ which models the expected reward if applying policy π when being in belief state b_t at time t. An optimal policy is then found by maximising V^{π} to yield V^* which is iteratively defined by

$$V^*(b_t) = \max_{a_t}[r(b_t, a_t) + \gamma \sum_{o_{t+1} \in O} P(o_{t+1}|b_t, a_t)V^*(b_{t+1})] \tag{2.8}$$

For an exact solution, the value iteration algorithm (Monahan, 1982) may be used. However, the algorithm becomes intractable already for a minimal configuration. Hence, other approaches have been investigated. Young et al. (2010) proposed a grid based method: only points within the summary space are stored and an optimal action is learned. During execution of the dialogue system, the current belief state is mapped into the summary space and the action of the closest grid point is executed. The complete algorithm is depicted in Algorithm 1. In the first part—the exploration phase—one dialogue is executed and at each turn, a random action is taken with some probability ε. The dialogue finally ends with a cumulative reward R. In the second part, the policy is updated: if there is a grid point already existing which is less then δ away from the grid point

Algorithm 1: Grid-based policy learning as published by Young et al. (2010)

Data: $\mathscr{B}$ a set of summary belief grid-points $\hat{b}_k$ in summary belief space, initial belief state b_0

Result: Updated policy $\pi(\hat{b}_k)$ for all grid-points within the summary belief space $\hat{b}_k$

repeat
- $t \leftarrow 0$
- $\hat{a}_0 \leftarrow$ initial greet action
- $b \leftarrow b_0$ `// all states in single partition`
- **repeat**
  - $t \leftarrow t+1$
  - Get user turn u_t and update belief state b
  - $\hat{b}_t \leftarrow$ toSummaryState(b)
  - $\hat{a}_t = \begin{cases} RandomAction & with probability \varepsilon \\ \pi(closest(\hat{b}, \mathscr{B})) & otherwise \end{cases}$
  - record $\langle \hat{b}_t, \hat{a}_t \rangle$
  - $T \leftarrow t$
- **until** *dialogue terminates with cumulative reward R from user simulator*
- `/* Scan dialogue and update` $\mathscr{B}$`, Q, and N */`
- **for** $t \leftarrow T$ **downto** *1* **do**
  - **if** $\exists \hat{b}_k \in \mathscr{B} : |\hat{b}_t - \hat{b}_k| < \delta$ **then**
    - $Q(\hat{b}_k, \hat{a}_t) \leftarrow \frac{Q(\hat{b}_k, \hat{a}) \cdot N(\hat{b}_k, \hat{a}_t) + R}{N(\hat{b}_t, hata_t) + 1}$
    - $N(\hat{b}_t, hata_t) \leftarrow N(\hat{b}_t, hata_t) + 1$
  - **else**
    - add $\hat{b}_t$ to $\mathscr{B}$
    - $Q(\hat{b}_k, \hat{a}_t) \leftarrow R$
    - $N(\hat{b}_t, hata_t) \leftarrow 1$
  - $R \leftarrow \gamma R$
- `/* Update policy */`
- **foreach** $\hat{b}_k$ *with updated* $Q(\hat{b}_k, \hat{a})$ *for any* $\hat{a}$ **do**
  - $\pi(\hat{b}_k) \leftarrow \arg\max_a Q(\hat{b}_k, \hat{a})$

until *converged*

at hand, the expected cumulative reward Q is updated. Otherwise, a new grid point is generated. Finally, based on the expected reward Q, the policy is updated.

While this grid-based approach is straightforward to implement, the downside of this learning technique is that hundreds of thousands of dialogues are needed as all relevant sub-spaces of the summary space must be populated with reference grid points. Another optimisation algorithm

based on Gaussian processes (GPs) (Engel et al., 2005) need less data but are more complex. Hence, within this thesis, we are more interested in applying policy-optimisation based on Gaussian processes.

GPs for their usage in SDS has been originally proposed by Gačić et al. (2010). The general idea is to use a GP for approximating the Q-function representing the expected cumulative reward Q given the summary state $\mathbf{c}$ and a system action a:

$$Q(\mathbf{c},a) = E_\pi(\sum_{\tau=t+1}^{T} \gamma^{\tau-t-1} r_\tau | c_t = \mathbf{c}, a_t = a) \; . \tag{2.9}$$

Here, r_τ is the reward obtained at time τ, T the length of the complete dialogue and $\gamma \in (0,1]$ a discount factor. Based on the Q-function, the system action a out of all system actions A may then be selected by

$$\pi(\mathbf{c}(b)) = \arg\max_{a \in A} Q(\mathbf{c},a) \; . \tag{2.10}$$

For estimating the parameters of the Gaussian distributions of the Q-function, a common method is using GP-SARSA depicted in Algorithm 2. Based on training data, the parameters are updated at time t using s_{t-1}, a_{t-1}, r_t, a_t, a_t (hence SARSA) having $s = \mathbf{c}$. However, for the update, all dialogue turns up to the current time t are needed and referenced to in a quadratic matrix which has to be inverted. This matrix inversion in the learning algorithm has a complexity of $\mathcal{O}(t^3)$. Thus, the GP-SARSA algorithm is computational intractable. Hence, Engel (2005) proposed an alteration based on kernel span sparsification which we will use within this thesis. The general idea is that instead of maintaining all previous turns, only relevant turns are stored in a dictionary. Based on this dictionary content, the probability distributions may be restored. To measure the relevance of a turn modelled as a state-action-pair (b,a), a kernel function $k((b,a)(a',b'))$ is used. Hence, for applying the learning algorithm for optimising a dialogue strategy, the kernel must be adjusted with respect to the dialogue domain. The full algorithm for its application in dialogue optimisation has thoroughly been described and evaluated by Gačić and Young (2014).

While the basic principles of IS and HIS have been presented, we will further introduce an actual implementation of a dialogue manager in the following section. It is based on the IS and will be extended within this thesis to also include HIS functionality (Section 5.2).

Information State Modelling with the OwlSpeak Dialogue Manager

To overcome the problems identified in Section 2.1.2 for the TrindiKit, the dialogue manager OwlSpeak has been created offering an alternative implementation of the IS paradigm. Heinroth et al. (2010a)'s OwlSpeak is based on the *model-view-presenter* design pattern (Potel, 1996) allowing for a strict separation of data management, dialogue logic and dialogue interface. By design, it allows for persistent dialogues which conveys multitasking functionality inherently (cf. Heinroth and Denich (2011)). Furthermore, OwlSpeak is implemented as a Java Servlet allowing for easy communication with speech devices. The general architecture is shown in Figure 2.6. In the following, the main aspects of OwlSpeak are briefly described. This is arranged as outlining the model, view, and presenter separately.

Algorithm 2: Episodic GP-SARSA policy learning as published by Gačić et al. (2010)

Data: A kernel function $k((b,a)(a',b'))$, an initial policy π based on an initial Q approximation
Result: Updated policy π based on an updated Q approximation

for *each episode* **do**
  Initialise b_t
  if *first episode* **then**
    Choose a_t arbitrarily
    $\mathbf{r}_t \leftarrow [\,]$
    $\mathbf{B}_t \leftarrow [(b,a)]$
    $\mathbf{K}_t \leftarrow [k((b,a),(b,a))]$
    $\mathbf{H}_t \leftarrow \begin{bmatrix} 1 & -\gamma \end{bmatrix}$
    Initialise Q-function
  else
    if *initial step* **then**
      Choose $a_t \leftarrow \pi(b_t)$ (using π derived from Q)
  for *each step in the episode* **do**
    Take action a_t
    Observe reward r_{t+1}
    Observe next state b_{t+1}
    if *non-terminal step* **then**
      Choose next action $a_{t+1} \leftarrow \pi(b_{t+1})$
      $\mathbf{B}_{t+1} \leftarrow \begin{bmatrix} \mathbf{B}_t & (b_{t+1}, a_{t+1}) \end{bmatrix}$
      $\mathbf{K}_{t+1} \leftarrow \begin{bmatrix} \mathbf{K}_t & \mathbf{k}_t(b_{t+1},a_{t+1}) \\ \mathbf{k}_t(b_{t+1},a_{t+1}) & k((b_{t+1},a_{t+1}),(b_{t+1},a_{t+1})) \end{bmatrix}$
      $\mathbf{H}_{t+1} \leftarrow \begin{bmatrix} \mathbf{H}_t & \mathbf{0} \\ \mathbf{u}^T & -\gamma \end{bmatrix}$, where $\mathbf{u} = \begin{bmatrix} \mathbf{0} & 1 \end{bmatrix}^T$
    else
      $\mathbf{B}_{t+1} \leftarrow \mathbf{B}_t$
      $\mathbf{K}_{t+1} \leftarrow \mathbf{K}_t$
      $\mathbf{H}_{t+1} \leftarrow \begin{bmatrix} \mathbf{H}_t \\ \mathbf{u}^T \end{bmatrix}$, where $\mathbf{u} = \begin{bmatrix} \mathbf{0} & 1 \end{bmatrix}^T$
    $\mathbf{r}_{t+1} \leftarrow \begin{bmatrix} \mathbf{r}_t & r_{t+1} \end{bmatrix}$
    Update Q-function
    if *non-terminal step* **then**
      Update $b_t \leftarrow b_{t+1}$
      Update $a_t \leftarrow a_{t+1}$

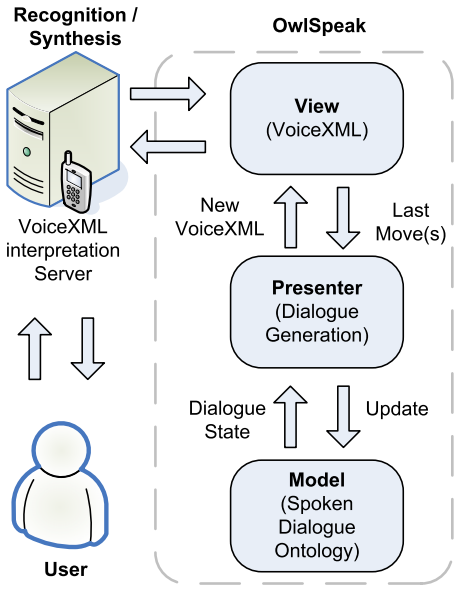


Fig. 2.6: The general architecture of OwlSpeak originally published by Heinroth et al. (2010a) following the model-view-presenter paradigm. OwlSpeak is implemented as a Java servlet providing VoiceXML output. By that, a voice browser may call the servlet and interpreting the returned VoiceXML document establishing communication between OwlSpeak and the user using Speech Synthesis and Recognition.

Spoken Dialogue Ontology (Model)

For designing the model of OwlSpeak, the Web Ontology Language (OWL) (Antoniou and van Harmelen, 2004) is used. In general, ontologies consist of *classes*, *relations* between classes, and instantiated class *individuals*. For dialogue management, a set of classes and relations has been predefined to establish spoken dialogue ontologies (SDOs). They contain both the static description of the dialogue as well as the current dialogue state. The schematic description is shown in Figure 2.7. The *Speech* part contains the static concepts of the dialogue. The five concepts are described in the following (note that the terms *class* and *concept* are used interchangeably):

Utterance The *Utterance* concept encapsulates one system utterance, i.e., one or more sentences the system may utter at one system turn. This represents the output of the system.

Grammar OwlSpeak uses grammars. The grammar belonging to one *Grammar* concept describes what input the user may provide and the system is able to understand.

Semantic A *Semantic* individual represents one information snippet important for the dialogue, i.e., the meaning of what was said by the system or the user.

Variable In contrast to Semantics, *Variables* are used for information which may take one out of several values provided during the dialogue, e.g., time information or destination. In addition, variables may also be used for system internal values.

Move The *Move* concept may either be a grammar move (system) or an utterance move (user) representing one atomic dialogue step. Furthermore, a grammar move represents the semantic representation of the user action. The *semantic* and *contrarySeman-*

tic relations define *Semantics* which are set, or unset respectively, when the move is performed.

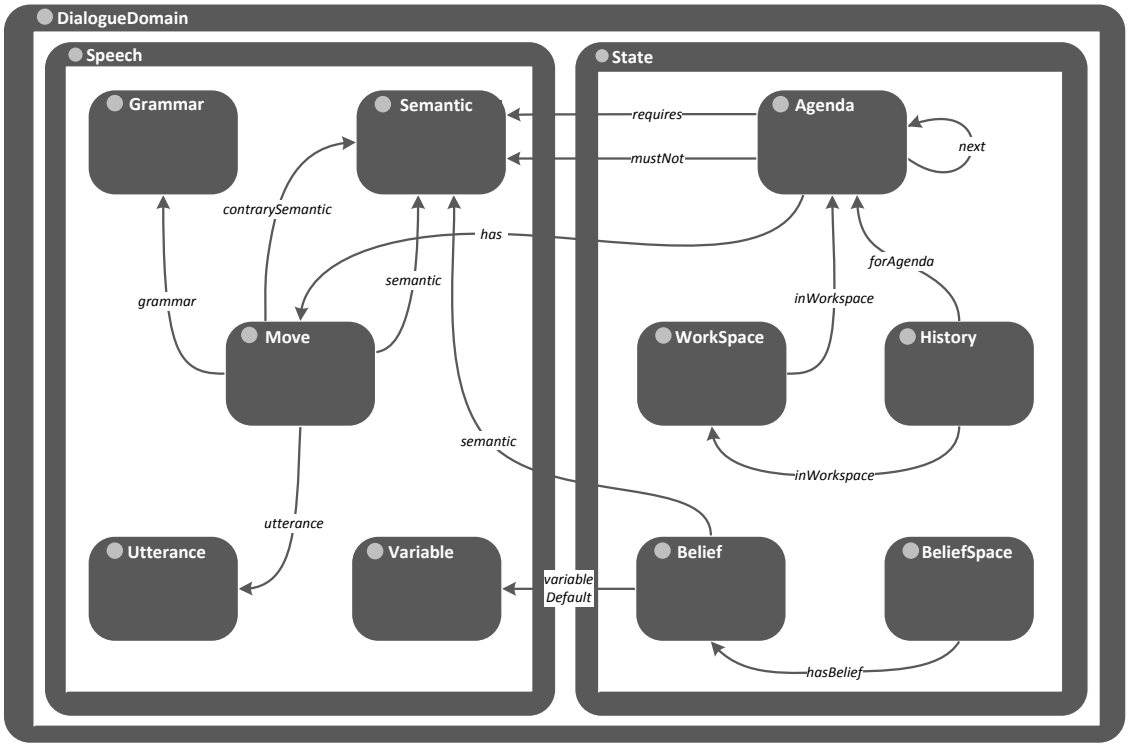


Fig. 2.7: A scheme of the Spoken Dialogue Ontology (SDO) originally published by Heinroth and Denich (2011). The static dialogue description is shown on the left side of the picture within the *Speech* class while the concepts belonging to the dynamic *State* of the system is shown on the right side.

By separating the utterances, grammars, and semantics from the moves, the individuals may be reused for other moves as well.

The current *State* of the dialogue system is stored in the dynamic part of the SDO. It is updated after each dialogue turn resulting in a persistent dialogue, i.e., after interrupting the dialogue, it may be picked up at the same position. The important concepts are described in the following while a more extensive description of the SDO also including the *History* concept can be found in (Heinroth et al., 2010b):

Belief While the semantics are static, a *Belief* represents a *Semantic* which is valid in the current dialogue state, i.e., if a move is performed, its semantics are transformed into a belief. Furthermore, as *Belief* may also represent a *Variable* with corresponding value valid in the current dialogue state.

BeliefSpace The *BeliefSpace* contains all instances of beliefs which are valid in the current dialogue state.

Agenda A system action is represented by the concept *Agenda*. In each system turn, an agenda is selected and executed. An agenda may contain zero or one utterance moves and several grammar moves. Furthermore, preconditions which have to be true are defined by the relations *requires* and *mustNot*. They define the *Semantics* which must (or must not) exist as a *Belief* in the *BeliefSpace* necessary for this agenda to be executed. Additionally, also *Variables* can be part of the preconditions. Furthermore, the *next* relation defines a set of agendas which are written into workspace after the agenda has been executed enabling their execution in following system turns.

WorkSpace A *WorkSpace* contains all agendas scheduled for execution. Note that the preconditions of those agendas do not have to be true in order to be scheduled. This remains the task of the policy.

Voice Document (View)

Communication between OwlSpeak and the input/output of the dialogue system (speech recogniser and speech synthesiser) is based on VoiceXML (Oshry et al., 2007). VoiceXML defines a set of tags which can be interpreted by a voice browser—comparable to HTML and a web browser—which in turn manages the links to ASR/NLU and TTS. It will be described in more detail in Section 3.2.1. OwlSpeak itself is implemented as a Java Servlet providing a new VoiceXML-document at each turn. The dynamically created document is based on the selected agenda where the utterance move provides the system prompt and all grammar moves are combined to form one big grammar.[6] By using the move names as a grammar tag, the user utterance can be mapped back to its corresponding move. Using a submit-tag, the input to the voice browser is passed back to OwlSpeak.

Dialogue Control (Presenter)

The dialogue control logic of OwlSpeak is located in the presenter. The entry point to the dialogue is defined by a flag of the *Agenda* class marking it as master agenda. Naturally, there may only exist one entry point per SDO. This master agenda contains in its *next* relation the initial set of agendas to be written to the workspace.

To select the next agenda to be executed, the preconditions of all agendas currently in the workspace are checked. If there is more than one agenda whose preconditions are fulfilled, the agenda is selected by relying on an additional priority score. The priority may be predefined and/or increase dynamically according to the amount of time the agenda is already in the workspace.

Based on the selected agenda, a view is created. It eventually passes new user input back to the presenter. In OwlSpeak, only the part of the user input which is necessary to identify the corresponding user move, i.e., the move name, arrives at the presenter. After determining the corresponding ontology, its belief space is updated by creating new *Belief* individuals and removing obsolete ones. Furthermore, the agenda is removed from the workspace while all agendas in the *next* relation are written into the workspace. Now the process starts anew: an agenda is selected, a view is created, and new user input is processed.

[6] Combining grammars may result in conflicts. This is taken care of by special conflict resolution algorithms which are not covered here.

Dialogue managers like OwlSpeak are designed to control the dialogue flow based on predefined strategies. These strategies are usually very domain specific. However, they incorporate general concepts which are briefly outlined in the following.

2.1.3 General Concepts of Dialogue Strategies

The strategy of a dialogue is the answer of the question "what is the best action to take given the current dialogue state and dialogue history?" Obviously, this is the same problem as outlined before for the dialogue manager. However, here, the problem is viewed from an interaction point of view (instead of the technical part).

The answer to this question is highly dependent on the actual task the SDS is designed to fulfill. However, some general concepts are common for most systems which are the *dialogue initiative* and *grounding* which will both be discussed in more detail in the following.

Dialogue Initiative

According to McTear (2004), the *dialogue initiative* models who of the two dialogue partners holds which role in the interaction between the user and the system, e.g., who is questioner and who is questionee. Hence, there are three categories:

System-directed For system-directed dialogues, the SDS is in control of the dialogue and asks questions to the user, e.g., to collect information. Here, the user has no flexibility on what to say or to change the course of the dialogue. By restricting the valid user inputs, classification errors may be minimised and the goal of the interaction is very likely to be achieved. However, the interaction is not very natural and user-friendly.

User-directed A dialogue which implements purely user-directed initiative has usually the character of a Q&A interaction: the user poses a question which the system answers. After the answer, the user may ask further questions. Hence, the user decides the topic of the dialogue and is in control while the system simply reacts to the user input. While the user is free to formulate any question whose topic is covered by the system, the user has no guidance at all, i.e., the system does not provide any clue to the user to deduce what is acceptable user input.

Mixed-initiative In a dialogue which implements a mixed-initiative, "either participant can take the initiative to ask questions, initiate topics, request clarifications" (McTear, 2004). Hence, both participants are able to influence the course of the dialogue. Naturally, this is closest to natural interaction. However, mixed-initiative dialogues are very complex and not easy to model.

Grounding

The concept of grounding describes the mechanisms which are used within the conversation to establish a common ground between the interaction partners. This is, e.g., to get confirmation for some piece of information which has previously been shared.

In an ideal dialogue system where each user input is understood correctly, this would not be necessary. However, like in human-human-interaction, sometimes, the user utterance is not

understood correctly, e.g., due to speech recognition errors. Here, the system may want to clarify if the information understood by the system is correct. A common practice for dialogue systems is to use confirmation prompts, i.e., restating the information and allowing the user to confirm or decline.

Here, two main differences regarding the confirmation strategy are known:

Explicit Confirmation If the system asks the user explicitly to confirm a piece of information, a separate system utterance is created, e.g., "Did you say West North Avenue?", allowing the user only to confirm or negate. In some systems, the user is further allowed to provide corrected information. However, this also depends on the implemented dialogue initiative.

Implicit Confirmation By implicitly asking for confirmation, the piece of information to be grounded is solely restated and embedded in a system prompt. This prompt's main focus is on some other relevant piece of information, e.g., the departure time in "So West North Avenue. When do you want to leave?". To negate, the user may say something like "that is wrong" or even provide corrected information. Each input which does not address "West North Avenue" and provides a valid response to the question of the departure time is then regarded as confirmation.

While implicit confirmations clearly allow for smoother dialogues, inexperienced users may not understand the mechanisms thus accidentally confirming something which has actually not been understood correctly. If every piece of information is confirmed explicitly, though, this may not happen. However, these dialogues are quite long and monotonous.

In order to identify which grounding strategy is more applicable for the given context—or how well a dialogue strategy performs in general—, the system should be evaluated. Hence, we give a brief overview over evaluation methods in the following.

2.1.4 Evaluation of Spoken Dialogue Systems

In order to compare different dialogue systems or dialogue strategies, measures for the evaluation of SDSs must exist. However, the best way to evaluate spoken dialogue systems is still unknown. Often times, objective criteria are used.

Within this thesis, three objective metrics are used to evaluate the dialogue performance: the average dialogue length (ADL), the dialogue completion rate (DCR) and task success rate (TSR). The ADL is modelled by the average number of system or user utterances per completed dialogue. A dialogue is regarded as being completed if the system provides a result—whether correct or not—to the user. Hence, DCR represents the ratio of dialogues for which the system was able to provide a result:

$$DCR = \frac{\#completed}{\#all} .$$

TSR is the ratio of completed dialogues where the user goal matches the information the system acquired during the interaction:

$$TSR = \frac{\#correctResult}{\#completed} .$$

Another way of evaluating SDSs is to get the user's opinion about the interaction. This is usually done using questionnaires. While there exist a multitude of standardised questionnaires (e.g. (ITU, 2003) for telephone-based speech applications or (Hassenzahl et al., 2003) for attractiveness and general usability of technical systems), the SASSI questionnaire (Hone and Graham, 2000) has emerged to be the quasi-standard for purely speech-based dialogue systems. It consists of 34 statements about the system and the interaction which may be rated on a Likert scale consisting of 5 or 7 ratings. The SASSI questionnaire has been designed to be applicable to a wide range of dialogue systems still providing easy means of collecting and evaluating the subjective user ratings. An example of the SASSI questionnaire is shown in Figure A.2. Moreover, Engelbrecht (2012) has provided a thorough discussion of the SASSI questionnaire in comparison to other questionnaires.

Naturally, collecting the user's opinion entails the necessity to conduct user studies with real users. Here, *real users* is emphasised as another option exists for evaluating dialogue systems without real users by applying user simulation techniques. Generally speaking, these user simulators are designed to mimic the user behaviour (including the variance induced by different users). While user simulation clearly has its limits as it will not be able to capture all variation in user behaviour, the obvious advantage is that thousands of dialogues may be performed easily without much effort in a short amount of time. In the simplest cases of user simulation, a statistical model is created based on a set of sample data to model the probability $p(s_u|s_a)$ of the user action a_u given the system action a_s. Of course, more elaborated approaches are also known, e.g., hidden agenda user simulation (Schatzmann and Young, 2009).

While we have provided a thorough overview over the relevant aspects of spoken dialogue systems, for rendering such systems user-centred, the user state must be derived. Here, statistical machine learning approaches will be used. In the next section, the approaches used in this thesis along with other relevant information regarding machine learning will be introduced.

2.2 Statistical Machine Learning Approaches

To render a spoken human-machine interaction user-centred, the system should be capable of changing its behaviour based on the user state. As the user state is not know, mechanisms are required which enable to estimate the user state. Here, machine learning techniques are used. While background on Spoken Dialogue Systems and all associated relevant aspects has been described in the previous section, we will give an overview over the machine learning in this section covering several different classification approaches.

Pattern classification according to Duda et al. (2001) is

the act of taking in raw data and making an action based on the "category" of the pattern.

Within this work, the categories or *classes* are the different user states. Designing and applying classification approaches comprises many different sub-tasks and sub-challenges. In general, the classification process starts with extracting a set of features from the raw data which are relevant to the classification task. The colour of a car, for example, is not relevant for classifying the make and model. Furthermore, the features may undergo a pre-processing and normalisation step (e.g., rotation, translation, scaling) to remove information irrelevant to the classification task.

Having defined the set of features, a classification model is created. Based on sample data, the model is designed to reduce the cost of misclassification. Here, the classification is associated

with a cost function defining the cost of estimating the class $\hat{\omega}$ for a sample while the true class is ω_c. Having this cost function, the model's classification decisions are based on minimising the total costs based on the training data. Thus, the model defines decision boundaries in the feature space effectively creating partitions having each associated with one of the target classes. Here, the problem is to find a decision function which minimises the costs not only on the training data but also on unseen data. This generalisation is usually tested by dividing the complete data into a training and evaluation set where the latter is only used for testing the classification model and its generalisation capabilities.

For training the model, i.e., learning from the sample data, three different methodologies may be applied according to Duda et al. (2001):

Supervised Learning	This method may be compared to learning with a teacher: for each sample to be learned on, the teacher provides the correct answer. To learn, the estimation may then be compared to the correct answer to improve the estimation model, e.g., using gradient descent.
Unsupervised Learning	Here, no teacher is present and hence the correct answer is unknown. Therefore, only natural categories may be found, i.e., using clustering approaches. The downside is that the clusters still lack a semantic interpretation: what class may be represented?
Reinforcement Learning	Similar to supervised learning, a teacher knows the correct answer. However, the correct answer is not provided but only information about whether the estimation is correct or not.

Within this thesis, only supervised learning and reinforcement learning are applied. Reinforcement Learning-based dialogue policy optimisation has been briefly touched in Section 2.1. For estimating the user state, supervised learning approaches are used. As the user state is modelled consisting of several categories, the classes are known. Thus, unsupervised learning approaches are not of interest.

Summarised, to solve a problem using statistical pattern classification, three questions must be answered: what is the best model to use, what are the best features, and does the training data cover all relevant aspects of the problem.

The field of supervised learning may be further divided into static and sequential approaches. While the former regards each sample data as an independent individual, sequential approaches model each sample belonging to a sequence of samples thus not being independent. Clearly, a human-machine dialogue consisting of subsequent system and user actions (or utterances) which depend on the previous actions thus comprising a sequence in the sense described above. On the other hand, handling each dialogue action as being independent thus reducing complexity may also be a valid approach. Hence, we have selected static and sequential approaches for our work which will be described along with methods for testing their performance in the following.

2.2.1 Static Supervised Classification

For static supervised classification, each data sample is regarded as not belonging to any data sequence. This is a very common problem thus many different classification approaches have been developed and analysed. For this work, though, we are focusing on approaches reflecting three different characteristics: probability modelling, linear discrimination, and rule learning.

Probability Modelling with Naïve Bayes

For modelling the classification problems using probability functions, the task may be described as finding the class $\hat{\omega}$ which maximises the posterior probability $P(\omega|\mathbf{o})$ given an observation vector $\mathbf{o}$:

$$\hat{\omega} = \arg\max_{\omega} P(\omega|\mathbf{o}) \; . \tag{2.11}$$

For deriving the posterior probability, the Naive Bayes classifier may be used. It calculates the posterior probability $P(\omega|\mathbf{o})$ of having class ω when seeing the n-dimensional observation vector $\mathbf{o}$ by applying Bayes rule (Duda et al., 2001):

$$P(\omega|\mathbf{o}) = \frac{p(\mathbf{o}|\omega) \cdot P(\omega)}{p(\mathbf{o})} \; . \tag{2.12}$$

In general, observations, i.e., elements of the observation vector, may be correlated with each other and introducing independence assumptions between these elements usually does not reflect the true state of the world. However, correlations are often not very high thus simplifying the Bayes problem has proved to result in reasonable performance for many problems. This is utilised by the Naïve Bayes classifier by assuming said independence thus calculating

$$p(\mathbf{o}|\omega) = \prod_{i=1}^{n} p(o_i|\omega) \; . \tag{2.13}$$

Thus, only the parameters of the n probability functions $p(o_i|\omega)$ have to be estimated based on the training data. For modelling these probability functions with Gaussians, for example, $2n$ parameters have to be estimated (the statistical means μ_i and the variances σ_i^2).

While the Naïve Bayes classifier is based on predefined models where only the optimal parameter configuration must be found, approaches based on linear discrimination will be described in the following.

Linear Discrimination with Artificial Neural Networks and Support Vector Machines

Linear discrimination is based on the idea of separating the classes by parametric linear decision functions $k_i(\mathbf{o})$ which are directly learned from the training data. Hence, no previous knowledge about the actual probability distribution is necessary. In Figure 2.8, an example is shown for a two-dimensional two class problem which may be separated by a single linear function $k(\mathbf{o})$. The decision for feature vector $\mathbf{o}$ is then made based on this decision function $k(\mathbf{o})$:

$$\hat{\omega} = \hat{\omega}(\mathbf{o}) := \begin{cases} \omega_1 \text{ if } k(\mathbf{o}) > 0 \; , \\ \omega_2 \text{ if } k(\mathbf{o}) < 0 \; , \\ \mathit{undefined} \text{ else} \; . \end{cases} \tag{2.14}$$

A widely-used approach to learn a linear discrimination decision function are multi-layer perceptrons (MLPs) (Rosenblatt, 1958) being a subset of artificial neural networks (ANN). The idea is to mimic the structure of the brain and create a set of connected artificial neurons. One neuron of such an MLP models a function k_j which linearly weights the elements of the input vector o_i and calculates the result by comparing the weighted sum to a threshold θ_j:

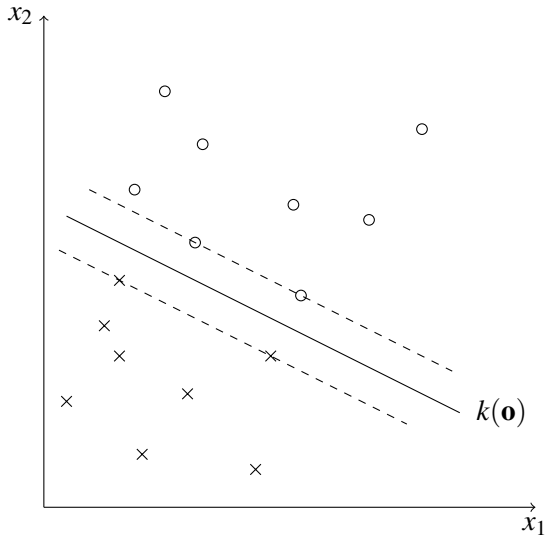


Fig. 2.8: Example for linear discrimination.

$$k_j(\mathbf{o}) = \begin{cases} 1 \text{ if } \sum_i \alpha_{ij} o_i > \theta_j \text{ ,} \\ -1 \text{ else .} \end{cases} \tag{2.15}$$

The final decision is then made by combining the output of these neurons within an output neuron. In this output neuron, the decision function $k(\mathbf{o})$ may be modelled, e.g., by

$$k(\mathbf{o}) = \sum_j w_j k_j(\mathbf{o}) + b \tag{2.16}$$

weighting all k_j with a weighting factor w_j and adding a bias b. The final decision for a two class problem may then be made as described in Equation 2.14.

A general graphical representation with one input layer, one hidden layer and one output neuron is shown in Figure 2.9. Of course, to allow for more complex decisions to be made, additional layers may be introduced between the input and output layer.

To learn the parameters α_{ij}, θ_j, w_j, and b from data, the back-propagation algorithm (Rumelhart et al., 1986) is used. The error between the estimated and the true value is calculated and propagated through the network to all neurons to optimise the parameters, e.g., using gradient descent.

A different approach for linear discrimination has been proposed by Vapnik (1995) called Support Vector Machine (SVM). For a two class problem, an SVM is based defining a hyperplane separating the two classes with maximum margin. The model is then defined by the support vectors which define the margin for both classes. This is illustrated in Figure 2.10 (using the same example for linear discrimination as in Figure 2.8) having the support vectors in bold.

The decision function $k(\mathbf{o})$ representing the hyperplane is then defined as

$$k(\mathbf{o}) = \sum_{i=1}^{N} \alpha_i z_i K(\mathbf{m}_i, \mathbf{o}) + b \text{ ,} \tag{2.17}$$

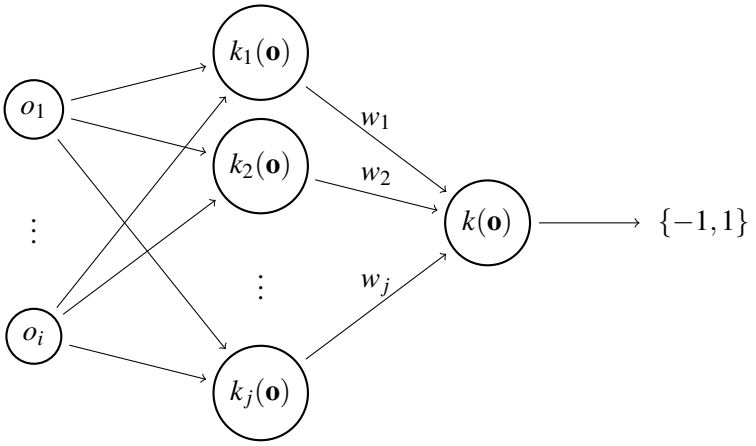


Fig. 2.9: The generic architecture of a multi layer perceptron.

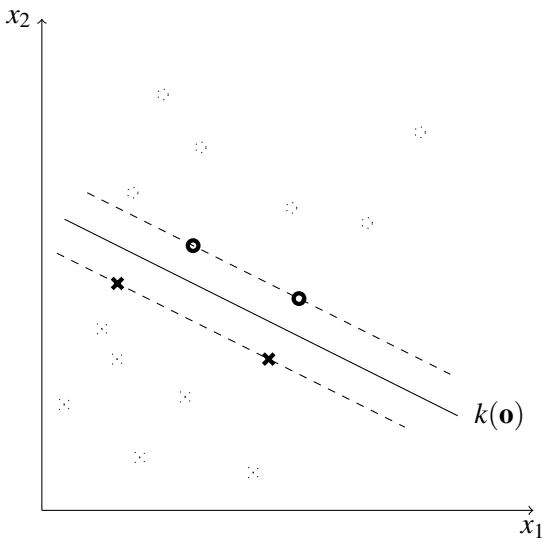


Fig. 2.10: Example for linear discrimination using a support vector machine. The support vectors are drawn in bold.

where $\mathbf{m}_i$ represent support vectors defining the hyper plane (together with b), z_i the known class $\mathbf{m}_i$ belongs to, α_i the weight of $\mathbf{m}_i$, and $K(\cdot,\cdot)$ a kernel function. The kernel function is defined as

$$K(\mathbf{m},\mathbf{m}') = \langle \varphi(\mathbf{m}), \varphi(\mathbf{m}') \rangle \,, \tag{2.18}$$

where $\varphi(\mathbf{m})$ represents a transformation function mapping $\mathbf{m}$ into a space Φ of different dimensionality and $\langle \cdot,\cdot \rangle$ defines a scalar product in Φ. By using the kernel function, the linear discrimination may happen in a space of high dimensionality without explicitly transforming the observation vectors into said space.

Again, the estimated class $\hat{\omega}$ for observation vector $\mathbf{o}$ is based on the sign of the decision function $k(\mathbf{o})$ as described in Equation 2.14.

According to Beyerer (2008), to train an SVM, the problem is stated as finding the configuration which minimises the equation

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle - \sum_{i=1}^{N} \alpha_i [z_i (K(\mathbf{m}_i, \mathbf{o}) + b) - 1] \tag{2.19}$$

using the method of Lagrange multipliers. All non-support vectors result in an α-value of 0.

A completely different approach to defining a discriminating hyperplane is to find a set of rules which will be explained next.

Rule Learning with the RIPPER Algorithm

Creating a classifications with rule learning is based on the idea of defining a set of rules to assign classes $\hat{\omega}$ to observation vectors $\mathbf{o}$. Additionally, a definition is needed on how the set of rules are applied. The RIPPER (Repeated Incremental Pruning to Produce Error Reduction) has been proposed by Cohen (1995) as a well-functioning and computationally efficient algorithm to define the set of rules. There, each rule consists of conjunctions of $A_n = v$, where A_n is a nominal attribute, or $A_c \geq \theta, A_c \leq \theta$, where A_c is a continuous attribute. An example rule for the attributes *colour* and *length* is depicted in the following:

$$\text{if } colour = green \text{ and } length \geq 3.2\text{m then } \hat{\omega} = \omega_1 \ .$$

Each part of the observation vector $\mathbf{o}$ is reflected by one of the attributes. The rules are then ordered and executed according to the following rule: unless a rule is evaluated successfully, move to the next rule; else apply the rule and stop.

The basic process of the algorithm for generating rules is divided into three steps: First, rules are grown by adding attributes to the rule. Second, the rules are pruned. If the resulting rule set is not of sufficient performance, all training examples which are covered by the generated rules are removed from the example set and a new rule is created.

Until now, probability modelling, linear discrimination and rule learning have been described as candidates for static classification. For regarding the problems of user state recognition in dialogues as a sequential problem, further algorithms may be considered.

2.2.2 Sequential Supervised Classification

While several approaches for static supervised classification have been presented where each data sample is regarded as being independent, sequential approaches regard each samples belonging to a sequence of data samples and thus being dependant of their preceding samples within the same sequence. For such problems, special classification algorithms have been developed. Within this thesis, a Hidden Markov Model (HMM) and a Conditioned HMM (CHMM) are applied. Both will be described in the following.

Hidden Markov Model

The idea of the Hidden Markov Model has initially been proposed by Rabiner (1989) for solving the problem of speech recognition (see Sec. 2.1).

Informally, an HMM consists of several states and—based on observations made—it models the progress through these states over time. However, the states may not be observed directly (the states are *hidden*). Instead, only uncertain observations of the current state are available. Given an unseen sequence of observations, the HMM may either be used to compute a probability of this sequence given the model parameters, which is called *evaluation*, or to find the most likely state sequence, which is called *decoding*. For the latter, each state is associated to a class. Speech recognition is an example for *decoding* where each state is tied to a phoneme.

Formally, an HMM is defined by the 5-tuple (S, O, A, B, π), where S is the set of states s, O the set of multi-dimensional observation vectors $\mathbf{o}$, A the transition matrix containing the probabilities $a_{ij} = p(\mathrm{s}^{(t)} = s_j | \mathrm{s}^{(t-1)} = s_i)$ of transitioning from state i to state j in time step t, B the matrix of probabilities $b_j(\mathbf{o}) = p(\mathrm{o}^{(t)} | \mathrm{s}^{(t)} = s_j)$ of observing $\mathbf{o}$ in state j at time t, and π the initial probability distribution over all states with $\pi_i = p(\mathrm{s}^{(1)} = s_i)$ being the probability of starting in state i.

An example model with three hidden states s_i is shown in Figure 2.11 illustrating the state transitions a_{ij}, the observation models $b_i(\mathbf{o})$, and the initial probabilities π_i.

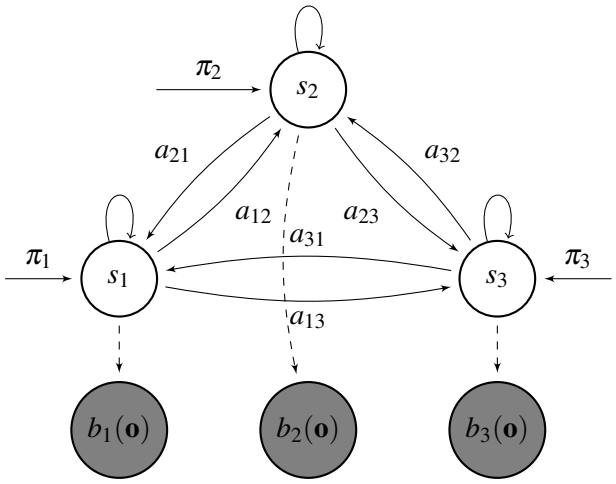


Fig. 2.11: This is an example of a HMM with three hidden states which are fully connected. Furthermore, each state has a connected observation probability model.

To compute the probability $p(\mathrm{o}|\lambda)$ of observing a sequence o for an HMM λ representing the *evaluation* task, the forward algorithm is used. For each time step t the current probability distribution over all states $\alpha_t(s_j)$ is iteratively computed by

$$\alpha_t(s_j) = \begin{cases} b_j(\mathrm{o}^{(1)}) \cdot \pi_j & \text{if } t = 1 \,, \\ b_j(\mathrm{o}^{(t)}) \cdot \sum_{i \in s} a_{ij} \cdot \alpha_{t-1}(i) & \text{else} \,. \end{cases} \tag{2.20}$$

The probability $p(\mathrm{o}|\lambda)$ is then computed by

$$p(\mathrm{x}^{(n)}|\lambda) = \sum_{i \in S} \alpha_T(i) \,. \tag{2.21}$$

To find the most likely state sequence for *decoding*, the Viterbi (1967) algorithm may be used. It is similarly defined to the Forward algorithm: instead of using the sum over all previous values, only the maximum is used.

For training of an HMM, Baum-Welch algorithm is usually used. It is based on the Forward-Backward algorithm (cf. (Rabiner, 1989)).

While the HMM may only be used for multi-class problems indirectly, the CHMM presented next accounts for that by allowing the estimation of a class probability directly.

Conditioned Hidden Markov Model

The Conditioned Hidden Markov Model (CHMM) is an extension of the classical HMM and has been originally published by Glodek et al. (2011). The main difference is that it directly introduces the concept of classes ω into the model. Hence, like the classical HMM, the CHMM also consists of a discrete set of hidden states $s_i \in S$ and a vector space of observations $\mathbb{O} \subseteq \mathbb{R}^n$. A separate emission probability $b_i(\circ^{(t)})$ is linked to each state defining the likelihood of observation $\circ^{(t)} \in \mathbb{O}$ at time t while being in state s_i. Further, $a_{ij,\omega} = p(\mathrm{s}^{(t)} = s_j | \mathrm{s}^{(t-1)} = s_i, \boldsymbol{\omega}^{(t)} = \omega)$ defines the transition probability of transitioning from state s_i to s_j. In contrast to the classical HMM, the transition probability distribution also depends on the class label $\omega \in \Omega$. This results in the transition matrix $\mathbf{A} \in \mathbb{R}^{|S| \times |S| \times |\Omega|}$. Furthermore, the meaning of the initial probability $\pi_{i,\omega} = p(\mathrm{s}^{(1)}) = s_i | \boldsymbol{\omega}^{(1)} = \omega)$ for state s_i is altered. It additionally represents the class probability for class ω at any time with the corresponding matrix $\boldsymbol{\pi} \in \mathbb{R}^{|S| \times |\Omega|}$. A schematic example of a CHMM with two classes and three hidden states is illustrated in Figure 2.12.

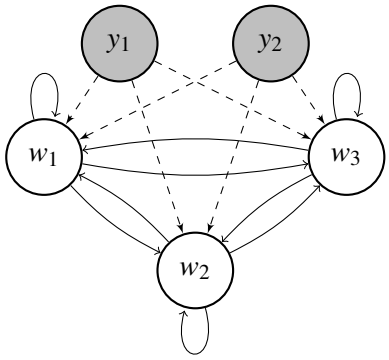


Fig. 2.12: This is an example of a CHMM with two labels and three hidden states originally published in (Ultes et al., 2012a). The dashed lines represent the label dependence of the hidden states, while the full lines illustrate state transitions. Please note that state transitions also depend on the labels which is not shown here.

In contrast to the classical HMM—computing the probability $p(\circ^{(n)} | \lambda)$ of observing the sequence $\circ^{(n)}$ thus multi-class recognition may only be performed by instantiating a model for each class separately or by tying classes to hidden states—, the CHMM allows for directly computing a class probability $p(\omega | \circ^{(n)}, \lambda)$. Similar to the evaluation of an HMM, this is done for the CHMM using a modified Forward algorithm by altering the definition of $\alpha_{t,\omega}$ (see Eq. 2.20):

$$\alpha_{t,\omega}(s_j) = \begin{cases} b_j(\circ^{(1)}) \cdot \pi_{j,\omega} & \text{if } t = 1 \ , \\ b_j(\circ^{(t)}) \cdot \sum_{i \in S} a_{ij,\omega} \cdot \alpha_{t-1,\omega}(i) & \text{else} \ . \end{cases} \tag{2.22}$$

Naturally, this alteration also entails the modification of the Baum-Welch algorithm used for training.

To identify the estimated class $\hat{\omega}$, the posterior probabilities $p(\omega|\circ^{(n)})$ are computed:

$$\hat{\omega} = \arg\max_{\omega} p(\omega|\circ^{(n)}) = \arg\max_{\omega} \frac{p(\circ^{(n)}, \omega)}{\sum_{\omega} p(\mathrm{x}\circ^{(n)}, \omega)} \ . \tag{2.23}$$

Here $p(\circ^{(n)}, \omega) = p(\circ^{(n)}|\omega)p(\omega)$, $p(\omega)$ is the prior probability over all labels and

$$p(\circ^{(n)}|\omega) = \sum_{s_i \in S} \alpha_{T,\omega}(s_i) \tag{2.24}$$

the probability for observation sequence $\circ^{(T)}$ given ω.

Until now, we have presented several static and sequential classification methods which we will apply for user state estimation. The goal in our thesis is to find out how the approaches perform for estimating the respective user state. To do this, we additionally rely on methods for evaluating these approaches. These evaluation methods will be describe in the following section.

2.2.3 Evaluation Methods for Classification

While several different classification approaches—static or sequential—may be applied for the same problem in general, the question remains which approach performs best. Here, *best* is quite relative as this highly depends on the applied evaluation metric. The most common metric is the accuracy—the ratio of correctly classified samples with respect to all tested samples. However, the accuracy does not account for unbalanced data, i.e., data which is not equally distributed over the classes. In a two-class example where 90% of the data belongs to class *a* and the rest belongs to class *b*, a simple classifier always assigning the majority class would result in an accuracy of 90% but would in fact not work at all. Hence, other metrics are within this thesis when dealing with unbalanced data. These metrics will be described in the following.

Another problem is the data which is used for training and evaluation. In general, training and evaluating on the same data is not advisable: the classification algorithm might simply "memorise" the correct answer. If the same sample is then used for evaluation, it is very likely to be classified correctly. However, these results do not show how the classifier would perform on unseen data. In other words, a classification algorithm should be tested with respect to generalisation. This aspect is also discussed within this section.

Evaluation Metrics

For evaluating the classification task within this thesis, we rely on four different metrics which all reflect different aspects. As outlined above, the *accuracy* is not suitable. A similar metric which regards effects of unbalanced data is the *Unweighted Average Recall* (UAR) or unweighted accuracy. The UAR is defined as the sum of all class-wise recalls r_c divided by the number of classes $|C|$:

$$UAR = \frac{1}{|C|} \sum_{c \in C} r_c \ . \tag{2.25}$$

Recall r_c for class c is defined as

$$r_c = \frac{1}{|R_c|} \sum_{i=1}^{|R_c|} \delta_{h_i r_i} \ , \tag{2.26}$$

where δ is the Kronecker-delta, h_i and r_i represent the corresponding hypothesis-reference-pair of rating i, and $|R_c|$ the total number of all ratings of class c. For the previous two-class example having a majority-class-classifier, the UAR would be 0.5 thus much better reflecting the weakness of the classifier.

Another metric originally developed by Cohen (1960) is *Cohen's Kappa*. It measures the relative agreement between two corresponding sets of ratings and may also be applied to evaluate classification tasks. This relative agreement is defined as the number of agreements corrected by the chance level of agreement divided by the maximum proportion of times the ratings could agree at all. Hence, κ is defined as

$$\kappa = \frac{p_0 - p_c}{1 - p_c} \ , \tag{2.27}$$

where p_0 is the rate of agreement and p_c is the chance agreement.

Of course, both UAR and Cohen's Kappa are very strict: both only reward exact matches and penalise all other estimates. While this behaviour is unproblematic for nominal classes, classes which have a natural order may be evaluated differently. Some estimates might be closer to the true estimates than others thus a different penalty might be better. Thus, Cohen (1968) introduced a weighting factor into the kappa formula allowing for weighted penalties based on the relative distance to the true class. Thus, the weighting factor w reduces the discount of disagreements the smaller the difference is between the two ratings:

$$w_{12} = \frac{|r_1 - r_2|}{|r_{max} - r_{min}|} \ . \tag{2.28}$$

Here, r_1 and r_2 denote the rating pair and r_{max} and r_{min} the maximal and minimal rating. This results in $w = 0$ for agreement and $w = 1$ if the ratings have maximal difference.

The weighting factor is then used within the computation of Cohen's Kappa:

$$\kappa = 1 - \frac{\sum_{i \in C} \sum_{j \in C} w_{ij} \cdot r_{ij}}{\sum_{i \in C} \sum_{j \in C} v_{ij} \cdot \frac{r_{\cdot i} \cdot r_{j \cdot}}{N}} \ . \tag{2.29}$$

Here, C is again the set of all classes and w_{ij} the weighing factor for estimating class i while the true class is class j. r_{ij} is the number of times i has been estimated while j is the true class. Thus, $r_{\cdot i}$ is the total number of times class i has been estimated, $r_{j \cdot}$ the total number of times the true class j is present in the data set of size N.

Another method for comparing classification results for ordinal classes is *Spearman's Rank Correlation Coefficient*. It measures the correlation between the reference classes and their estimates or, more general, the correlation between two ordinal variables. The correlation of two variables describes the degree by that one variable can be expressed by the other. *Spearman's Rank Correlation Coefficient*, or short *Spearman's Rho* is a non-parametric method assuming a monotonic function between the two variables (Spearman, 1904):

$$\rho = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}} , \tag{2.30}$$

Here, x_i and y_i are corresponding ranked ratings and $\bar{x}$ and $\bar{y}$ the mean ranks. Thus, two sets of ratings can have total correlation even if they never agree. This would happen if all ratings are shifted by the same value, for example.

In cases where a numerical representation of the classes exist, another evaluation metric is the root mean squared error (RMSE) usually applied in regression tasks. By calculating the error and comparing each estimated class $\hat{\omega}_i$ with the reference r_i—the known ground trough—for all N evaluation samples, RMSE is defined as

$$rmse = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\hat{\omega}_i - r_i)^2} . \tag{2.31}$$

While four different evaluation metrics which will be used within the thesis to evaluate the user state recognition approaches have been presented, a reasonable separation of the data is also very crucial for generating valid results. A very common approach to separate the data into different sets for training and evaluation will be described in the next section.

Generalisation and Cross-Validation

For creating statistical classification models, usually, only a limited amount of annotated data is available. However, for most algorithms, the more data you have the better the trained model will perform and generalise on unseen data. The latter is very crucial as, to test the performance of the classification method, unseen data is of special importance as argued before. This is, as the goal of classification is to find a model which is able to find the correct class with minimal error on unseen data. A common method is to statically split the data into a training and evaluation set, e.g., using 70% of the data for training and testing its performance on the remaining 30%. The downside of statically splitting is that, usually, random splits are used and neither of both sets may reflect the data distribution of the complete data set.

One way to remedy this effect is to use a method called m-fold cross-validation. Here, the data is split into m partitions of equal size. For evaluation, one partition is selected and the classifier which was trained on the remaining $m-1$ partitions is evaluated on this selected partition. Next, another partition is selected which has not been used for evaluation before. Again, the newly trained classifier is evaluated on this partition. This is repeated until all partitions have been used for evaluation. An example for this partitioning is shown in Figure 2.13 for 5-fold cross-validation. In the shown situation, P_2 has been selected for evaluation. Hence, the classifier will be trained on partitions P_1, P_3, P_4, and P_5.

To get a final performance value, one way is to calculate the performance for each fold separately and then taking the average of these results for each fold. Another way to get a final performance value is to store the results for each fold and calculate the performance measure afterwards using the complete data.

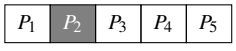


Fig. 2.13: An example for 5-fold cross-validation: partition P_2 is selected for evaluation while the remaining partitions are used to train the statistical model.

2.3 Summary of Relevant Background

For this chapter, our goal was to provide all background information necessary in order to understand the work which will be described in the content of this thesis. Knowing about the general modular pipeline architecture of a spoken dialogue system as well as the general operation of the single modules is important to understand how the human-computer interaction is executed and how the user state may be taken account of. For adapting to the user state, especially the different dialogue strategy concepts are also of interest. Finally, in order to understand the dialogue management approaches (and their implementation) for introducing adaptivity to the user state, the information state and hidden information state approaches have been explained. The extension of the presented approaches will be described in Chapter 5 along with experiments on user-adaptive dialogue modelling.

For deriving the user state automatically, various machine learning algorithms or statistical classification methods will be applied. In this chapter, we have described our selection of classifiers used to detect the user state automatically along with methodologies and measures to evaluate their performance. In Chapter 4, these classifiers will be employed to four different user states which depicts the first step to enable user-adaptive dialogue modelling.

While we have only provided an overview over the relevant aspects of both fields, further reading on spoken dialogue system technology has been presented by McTear (2004) an in-depth explanation of all relevant aspects. For further reading on statistical machine learning, we would like to refer to Duda et al. (2001).

Before we continue with user state recognition, we would like to introduce related work on both fields—user state and adaptive dialogue—and explain in detail the differences and the added value of our own work.

3

Related Work

The overall goal of the presented work in this thesis is to render human-machine dialogue more user-centred by incorporating information about the user into the dialogue manager. As has been described before, this goal may be divided into two problems: automatically recognising the user state and adapting the ongoing dialogue. Naturally, we are not the first having pursued this goal and, actually, many researchers have worked on either of the two fields. Hence, while we have laid the basis for understanding the contents of this thesis in the previous chapter, we will continue in this chapter by presenting related work in the two fields of user state recognition and user-adaptive dialogue management.

In order to adapt the dialogue to the user's state, the state has to be estimated first. Following this natural order, we will start with presenting the user states which we focus on in this thesis, describe the work of other research groups on how to automatically recognise them and explain the differences between their work and ours. Subsequently, we present related work on adaptive dialogue modelling by either extending the dialogue state or using the user information within the reward function of reinforcement learning approaches and again state the fundamental differences rendering our work as novel.

3.1 User State Recognition

For adapting the dialogue, a multitude of user states may be taken into account. Schmitt and Minker (2013) have identified several user states which are suitable for adaptation and grouped them into dynamic and static user states. Static user states describe states which do usually not change over the course of a dialogue, e.g., age, gender, or preferences. While many researchers have focused on these static properties (Schmitt et al., 2009a; Metze et al., 2007), in this thesis, we focus on dynamic user states. Adapting to a static state like age, for example, the dialogue may be tailored for different age groups by pre-selecting according dialogue strategies. However, for adapting to dynamic user states which will change during the interaction, more elaborated adaptation mechanisms are necessary to provide the flexibility needed. And for the system to deal with theses fluctuations and changing values during the dialogue, simply pre-selecting a suitable strategy is not sufficient.

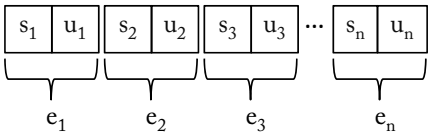
In this work, we have selected four dynamic user states: user satisfaction, perceived coherence, emotion, and intoxication. These are ordered according to their relevance to the dialogue

interaction. For virtually all dialogues, the user may be satisfied or unsatisfied with the interaction. As this is a general concept, most of our work will focus on *user satisfaction* (US) both for recognising US as well as using US for influencing the dialogue flow. Next, the *perceived coherence* of the system behaviour is considered. Coherence is also very general potentially occurring in all interactions: whether the user perceives the system's reactions, i.e., the system's behaviour as being coherent. A bit less frequently occurring in human-machine interaction are *emotions*. While those play without question a very important role in human-human interaction, still, machines are not yet capable of providing this fine-grained level of interaction capabilities. For some scenarios, adapting to emotions may be very beneficial for an SDS, e.g., cheering up sad users in a counselling dialogue. Finally, a very specialised (or exotic) user state is *intoxication*. Obviously, its potential application is limited to dialogues where the intoxication level is of interest, e.g., if the user is about to operate a motorised vehicle.

In the remainder of this section, we will introduce related work on the four before mentioned user states having in mind their application within the domain of spoken dialogue interaction.

3.1.1 User Satisfaction Recognition

User satisfaction in spoken dialogue represents a very general state occurring in almost all types of dialogue. Hence, many researchers have been involved in creating means of automatically identifying user satisfaction in spoken dialogue systems. Here, the work may be divided into two major directions: dialogue-level user satisfaction and exchange-level user satisfaction. A system-user exchange is defined as one unit of the dialogue. Hence, a dialogue consists of a sequence of system-user-exchanges which defined as one system turn followed by one user turn[1]:



Dialogue-level user satisfaction deals with the question whether the user is satisfied with the complete dialogue after the interaction has finished. User satisfaction on the exchange-level or turn-level, however, deals with the question of how the user is satisfied during the interaction with the dialogue up to the current point.

For creating models identifying the satisfaction of the user automatically in a supervised learning manner, data is needed which is annotated with the ground truth. Hence, how this ground truth is created is a problem which is diametrical to exchange-level vs. dialogue-level user satisfaction. The US annotations may be collected in general either by

- **users** during or right after the dialogue or
- **experts** by listening to recorded dialogues.

Here, users or *user raters* are people who actually perform a dialogue with the system and apply US ratings (single value ratings or questionnaires) while doing so. There is no constraint about their expertise in the field of human-computer interaction or Spoken Dialogue Systems: the users may be novices or have a high expertise. With experts or *expert raters*, we refer to people

[1] Figure originally published in (Schmitt et al., 2012)

who are not participating in the dialogue thus constituting a completely different set of people. Expert raters listen to recorded dialogues after the interactions and rate them by assuming the point of view of the actual person performing the dialogue.

For *User* Satisfaction, ratings applied by the users seem to be clearly the better choice over ratings applied by third persons. However, determining true User Satisfaction is only possible by asking real users interacting with the system. Ideally, the ratings are applied by users talking to a system employed in the field, e.g., commercial systems, as these users have real concerns.

For such Spoken Dialogue Systems, though, it is not easy to get users to apply quality ratings to the dialogue—especially for each system-user-exchange. The users would have to rate either by pressing a button on the phone or by speech, which would significantly influence the performance of the dialogue. Longer dialogues imply longer call durations which cost money. Additionally, most callers only want to quickly get some information from the system. Therefore, it may be assumed that most users do not want to engage in dialogues which are artificially made longer. This also inhabits the risk that users who participated in long dialogues do not want to call again. Therefore, collecting ratings applied by users are considered to be expensive. One possible way of overcoming the problem of rating input would be to use some special installation which enables the users to provide ratings more easily (cf. (Schmitt et al., 2011b)). However, this is also expensive and the system's usability would be very restricted. This setup could most likely only be used in a lab situation.

Expert raters, on the other hand, are able to simply listen to the recorded dialogues and to apply ratings, e.g., by using a specialised rating software. This process is much easier and does not require the same amount of effort needed for acquiring user ratings. Furthermore, as already pointed out, we refer to experts as people who have some basic understanding of dialogue systems but are not required to be high-level experts in the field. That is why we believe that these people can be found easily.

Naturally, this lead to a trade-off decision for creating the ground truth between users and experts. In this section, we will give examples for both expert and user ratings. And while we focus on exchange-level user satisfaction in this thesis, we will start with presenting work on dialogue level user satisfaction with the PARADISE framework.

Dialogue-level User Satisfaction Recognition

Fundamental work on the US recognition after the interaction has been presented by Walker et al. (1997, 1998b, 2000) with the PARADISE (PARAdigm for Dialogue System Evaluation) framework, which is—to our knowledge—the first work dealing with the automatic recognition of user satisfaction.

PARADISE is based on a simple concept: information about the dialogue costs and the task success is used to create a linear regression model which targets user satisfaction. This general structure is depicted in Figure 3.1: with the ultimate goal of maximising user satisfaction, the task success rate is also maximised while minimising the associated costs of the dialogue at the same time. In the following, all measures will be described:

Task success For maximising user satisfaction, task success should also be maximised. It is measured using the kappa function already presented in the context of evaluating statistical classification (Sec. 2.2.3):

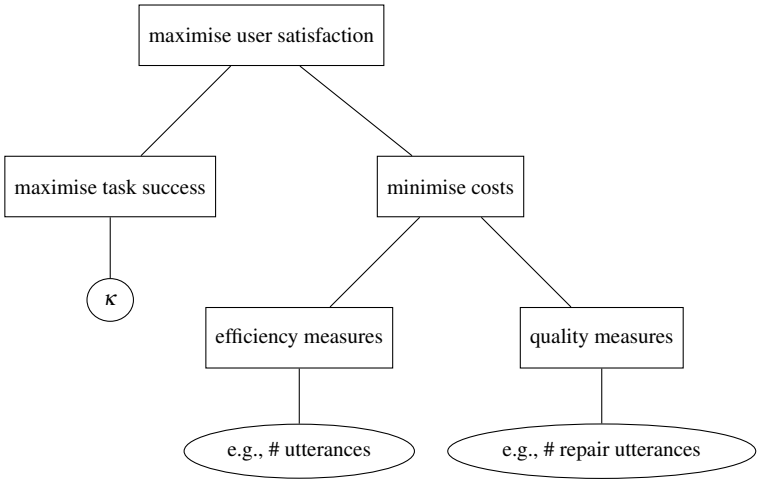


Fig. 3.1: PARADISE's structure of objectives as in (Walker et al., 1997).

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)} \,. \tag{3.1}$$

Again, $P(A)$ represents the probability of agreement and $P(E)$ the probability of agreeing by change. The *k*-function is applied to a confusion matrix for attribute-value pairs. Within this matrix, collected information of real dialogues are compared to predefined optimal dialogue scripts. By calculating the kappa value, the overall agreement between these two may be calculated.

Costs To minimise the costs of the dialogue, both efficiency measures and quality measures are considered. Efficiency measures are a straight forward concept, e.g., the number of utterances per dialogue representing the dialogue length. Then, cost c_1 would represent the dialogue length. For quality related measures, e.g., the number of repair utterances c_2 may be used in a similar manner.

For calculating a final user satisfaction measure *us*, both task success and costs are combined. To achieve a maximum value *us*, the costs, which should be minimised, are subtracted from the task success, which should be maximised:

$$us = (\alpha \cdot \mathcal{N}(\kappa)) - \sum_{i=1}^{n} w_i \cdot \mathcal{N}(c_i) \,, \tag{3.2}$$

where $\mathcal{N}(\cdot)$ represents a normalising function and α and w_i weighting factors. These weights have to be learned using linear regression. Here, the user's satisfaction with the dialogue is collected using questionnaires and used as target variable for the regression model.

Evanini et al. (2008) proposed a different measure for dialogue-level user satisfaction which they call "caller experience" (CE). They use a decision tree (Quinlan, 1992)—a concept similar to rule learning (cf. Sec. 2.2.1)—which uses parameters about the dialogue to automatically classify the caller experience. In a telephone-based hot-line setting, these parameters where chosen to be the "most informative in determining the CE" (Evanini et al., 2008): "the classification status of the call (how well the system determined the reason for the call), the number of speech recognition

errors during the call, the number of operator requests from the caller, and the exit status of the call (whether the callers task was completed, or where the caller was subsequently transferred)." While this setup seems to be similar to PARADISE where also a classificator is trained to identify the user satisfaction based on dialogue parameters, the big difference lies in the target variables: while for PARADISE, the user satisfaction is acquired using questionnaires, the caller experience is annotated by expert raters. This allows for using real dialogue data. The expert raters annotated each dialogue of the training set with a value ranging from one to five where five depicts the best rating. Their setup achieved an average agreement between the learned system and the rater annotations of $\kappa = 0.8$ using Cohen's weighted kappa as described in Section 2.2.3.

Unfortunately, both approaches are quite domain-dependent. For generating the regression model in PARADISE, weighting factors have to be computed for each system anew (Möller et al., 2008). This generates high costs as dialogues have to be performed with real users where each user further has to complete a questionnaire after completing the dialogue. While there is no information about the cross-domain capabilities of CE, it is very likely that the decision tree must be learned anew for each domain. Moreover, for rendering the dialogue system adaptive to the user's satisfaction, dialogue-level approaches are not suitable: the decision of how to continue the dialogue have to be made within each system-user exchange. Hence, we will present work on exchange-level ratings in the following.

Exchange-level User Satisfaction Recognition

Measuring the user satisfaction on the exchange-level means that anytime the system makes a decision about the next system action, the user satisfaction is determined anew. In this setting, using questionnaires is obviously not applicable. This is why only single-rated values are used. The following work is grouped by the way these values are applied: either by the users themselves or by experts.

Work comprising user ratings directly has been presented by Engelbrecht et al. (2009). They used Hidden Markov Models (HMMs) (see Sec. 2.2.2) to model the SDS as a process evolving over time. User Satisfaction was predicted at any point within the dialogue on a five-point scale comprising the ratings *bad*, *poor*, *fair*, *good*, and *excellent*. Each state of the HMM was tied to one of these five values. The progress through the model was based on observing six dialogue events like understanding errors. In a Wizard-of-Oz setting, six dialogues in the BoRIS domain (Bochum Restaurant Information System (Möller, 2005)) were scripted to ensure similar conditions for each participant. The participants applied user satisfaction ratings during the dialogue using a number pad after each turn. The dialogue was on hold until they finished rating. Evaluation was performed based on these user applied labels. For unseen data of two scenarios, the HMM achieved a performance of an average mean squared error of 0.086 and 0.037.

Hara et al. (2010) derived turn level ratings from an overall score applied by the users after the dialogue. The users applied ratings on a five-point scale which has been extended for the recognition task by indicating unsuccessful dialogues leading to six classes. Within the domain of a music retrieval system, they used n-gram models reflecting the dialogue history to estimate the user satisfaction after each turn. By testing different values for n, they achieved best performance for tri-grams ($n = 3$) which was only hardly above chance.

Work by Schmitt et al. (2011b) deals with determining User Satisfaction from ratings applied by the users themselves during the dialogues. Within the domain of Let's Go, a bus schedule

information system, the users performed predefined dialogues in the lab. During the interaction, each user applied user satisfaction ratings on a 5-point scale by indicating an increase or decrease in their satisfaction using a remote controller. Based on interaction parameters capturing important events and aspects of the interaction[2], a Support Vector Machine to automatically predict the user satisfaction for each turn achieving an MR/R of 0.49.

User ratings, though, have critical downsides: they are subject to strong variations and biased judgements. Furthermore, the problem of collecting these ratings exist: usually, only artificial dialogues are reasonably used to collect this data[3]. These problems are solved by using expert raters. They are able to listen to recorded dialogues of real world dialogue systems and have a more objective view on the dialogue. With this regard, related work on user satisfaction annotated by experts will be presented in the following.

Using experts instead of users for applying the user satisfaction ratings, Higashinaka et al. (2010a) proposed a model to predict turn-wise ratings for human-human dialogues (transcribed conversation) and human-machine dialogues (text from chat system). Ratings ranging from 1-7 were applied by two expert raters labelling "Smoothness", "Closeness", and "Willingness" not achieving a Match Rate per Rating (MR/R)[4] of more than 0.2-0.24 applying Hidden Markov Modes as well as Conditioned Random Fields. These results are only slightly above the random baseline of 0.14. Further work by Higashinaka et al. (2010b) used ratings for overall dialogues to predict ratings for each system-user-exchange using HMMs. Again, evaluating in three user satisfaction categories "Smoothness", "Closeness", and "Willingness" with ratings ranging from 1-7 achieved best performance of 0.19 MR/R.

Further work on user satisfaction recognition using expert annotations has been presented by Schmitt et al. (2011a) with their Interaction Quality (IQ). They connected interaction-related events to user satisfaction resulting in more robust and overall better performance than all the approaches presented before. Their general idea is to automatically derive a set of interaction parameters for each system-user-exchange and to use these as input to a statistical classifier which predicts the user satisfaction. Hence, each system-user exchange was annotated by three different expert raters using strict guidelines. These ratings ranging from 1-5 are used as target variable to train a support vector machine. They achieve a MR/R of 0.58. El Asri et al. (2014) applied ordinal regression thus exploiting the ordinal nature of the Interaction Quality ratings. On the same problem, they were able to further improve the recognition performance.

Based on the good performance results of the Interaction Quality and the benefit of using expert raters, our work on user satisfaction recognition will also be based on IQ. Thus, the Interaction Quality paradigm will be presented in detail in Section 4.1.1. While Schmitt et al. (2011a) and El Asri et al. (2014) already provided satisfying results modelling the problem as a static classification approach, their approaches lack in taking account for the temporal aspects of the problem: the user's satisfaction does not change suddenly but depends on how satisfied the user has felt before within the same interaction. Therefore, we will focus on these temporal aspects. For this, we will propose a plug-in sequential recognition architecture. There all types of state-of-the-art static classification approaches may be plugged-in. Finally, we will also compare this novel approach to state-of-the art static and sequential classifiers

[2] The parameters are the same as for the Interaction Quality paradigm described in Section 4.1.1

[3] For more details on user ratings for user satisfaction, please refer to Section 4.1.

[4] MR/R is equal to Unweighted Average Recall (UAR) which is explained in Section 2.2.3.

Aside from the user's satisfaction level, we are also considering three other user states. Here, the perceived coherence of the system behaviour is doubtless very important. Hence, approaches on coherence recognition will be presented in the following.

3.1.2 Perceived Coherence Recognition

Creating coherent dialogues is a very important task, not only because there is a relationship between the coherence of system actions and the degree of the user trusting the system (Muir, 1994). However, for identifying whether the dialogue partner views a conversation with a system as coherent, only limited related work exists. Gandhe and Traum (2008) presented a general study on how different models of dialogue coherence correlate with human judgments. They suggest to use n-gram statistics that are sensitive to linguistic features for modelling coherence. Their objective was to find an automatic mechanism that reduced the effort of human annotation to evaluate dialogue systems. Moreover, when the calculation of coherence is done not only once for each complete dialogue but consecutively throughout the dialogue for each turn, this information may also be used to change the course of the ongoing interaction.

In contrast to the general notion of discourse coherence where a complete dialogue is considered as being coherent or non-coherent (see, e.g., (Prakken, 2005; Purandare and Litman, 2008; Gandhe and Traum, 2008)), we are more interested in identifying coherence for each single system reaction. Furthermore, our aim is to be able to use coherence information for dialogue management, so we are interested in using dialogue acts information rather than linguistic features in order to make our approach general and suitable for any application domain. Hence, in contrast to the general notion of discourse coherence where a complete dialogue is considered as being coherent or non-coherent, we focus on analysing the coherence of each system dialogue act separately.

With the same aim, Noh et al. (2011) presented an approach to compute dialogue act coherence taking into account the dialogue structure. They introduce a measure called Discourse Coherence Indicator (DHI) for dialogue acts based on perplexity used to rank dialogue acts. Based on a sequence of dialogue acts, their approach takes into account task completion with different ASR error levels. While Noh et al. (2011) mainly base their approach on the dialogue sequence, interaction-related events may also strongly influence the coherence. Hence, our approach is not based on the actual dialogue sequence, but on the information provided by the user during the conversation and the live performance of relevant modules such as ASR and NLU.

While user satisfaction and coherence are more interaction-related user states, the user's emotion or intoxication may be derived from the speech signal directly. Here, the emotional state of the user has a long-standing research history. In the following, we will give a brief overview over relevant related work on speech based emotion and intoxication recognition.

3.1.3 Speech-based Recognition of Emotion and Intoxication

As emotions have shown to actually occur in human-machine spoken dialogue interaction (Reeves and Nass, 1996; Schmitt and Minker, 2013), we are also considering the emotional state of the user. However, several different ways of defining the term emotion exist. Plutchik (1980) define a set of eight emotions arranged in a circle (emotion wheel). Others rely on the "big six" emotions happiness, anger, disgust, sadness, surprise, and fear (Cornelius, 1996). While both previous definitions categorise and label the emotions, the pleasure arousal dominance (PAD) (Mehrabian,

1996) scale aims at creating a three-dimensional emotion space. Hence, occurring emotions need not be labelled but may simply be represented as a point within this space. Work on automatically estimating this point has been presented, e.g., by Grimm et al. (2007).

Our work, though, is focusing on emotion categories and this is already a hard problem. For approaches on automatically classifying emotions, one problem is to gather the training and evaluation data. Here, the distinction can be made between acted corpora, i.e., the emotions have been created by professional or amateur actors, or corpora containing real emotions. For the former, Schuller (2006) and Pittermann et al. (2009) have provided experiments on the Berlin corpus (Burkhardt et al., 2005) and on the Danish emotional speech database (Enberg and Hansen, 1996). For spontaneous emotions, a challenge has been organised by Schuller et al. (2009a) on the AIBO corpus (Batliner et al., 2004).

While the above stated work deals with emotion recognition in general, other people have laid the focus on speech-based telephone services, so called interactive voice response (IVR) systems. Petrushin (1999) recognised emotions of non-professional actors. Furthermore, Lee et al. (2001, 2002), Lee and Narayanan (2005), Batliner et al. (2000), Schmitt et al. (2010a), and Polzehl et al. (2011) presented work on recognising emotions in a deployed IVR system.

All classification approaches have in common that they use features extracted from the speech signal directly or features containing linguistic information (e.g., (Schmitt et al., 2010a; Polzehl et al., 2011; Lee et al., 2002)). However, as emotions are very personal, we aim at personalising the emotion recognition process by adding a speaker identification module to the overall recognition process. While a similar experiment adding the gender has already shown to increase the recognition performance (Vogt and André, 2006), we believe that a further personalisation yields in an even better overall recognition performance. The complete approach is described in Section 4.3.

A similar problem to emotion recognition represents the problem of automatically determining the user's intoxication state. While intoxication clearly poses a different target than emotions, approaches for both are based on extracting information from the speech signal and using it to train a statistical classification algorithm. In this context, Schuller et al. (2011) started the paralinguistic challenge at the Interspeech conference in 2011 dealing with intoxication recognition. There, a multitude of different approaches have been presented. Some modified the feature set (Bocklet et al., 2011; Bone et al., 2011; Hönig et al., 2011) for static classification approaches while others applied Hidden Markov Models (Nogueiras Rodríguez, 2011) for intoxication recognition or used multiple classifiers in a fusion-based approach (Montacié and Caraty, 2011).

While all presented approaches perform better or worse for automatically recognition the intoxication level of the user, a human baseline has not been defined yet. Hence, in our contribution to the challenge (Ultes et al., 2011b), we were aiming at comparing a standard static classification approach with the performance of human raters. While Schiel (2011) have presented a similar experiment in a lab situation, we present work on human performance in a more realistic scenario. This will be elaborated on in Section 4.4.

So far, we have presented the four user states we will consider in this thesis and have presented related work on automatically recognising those user states. As the general intention of our work is to render the dialogue user-centred, we will continue with describing work on user-adaptive dialogue management in the following.

3.2 User-Adaptive Dialogue Management

For rendering the dialogue manager user-adaptive, the user state should first be recognised automatically and then used in the dialogue manager to alter the course of the dialogue. While we have presented work on the former in the previous section, we will continue with describing the related work in the field of adaptive dialogue management.

Adaptive dialogue spans over many different types of adaptation. While some systems adapt to their environment (e.g., (Heinroth et al., 2010a)), the focus within this work lies on systems which adapt to the user and their characteristics. Here, a distinction can be made for static adaptation and dynamic adaptation. Static adaptation is based on a user model that contains static information which does not change over time, e.g., the user's age or the cognitive capabilities (Müller and Wasinger, 2002). However, within this work, an emphasis is placed on dynamic adaptation to the user during the ongoing dialogue (where the user state will likely change during the dialogue). We have identified two major research streams and grouped the related work accordingly: extending the dialogue state and modelling the reward function in a reinforcement learning setting.

3.2.1 Adaptation through Dialogue State Extension

One way to render a dialogue management system adaptive to the user state is to extend the dialogue state with the respective user state. Commonly, the dialogue strategy may then be modified in a rule-based manner (e.g., (Nothdurft et al., 2012; Gnjatović and Rösner, 2008; Litman and Pan, 2002; Conati and Maclaren, 2005)). However, some approaches also consider the dialogue system as a POMDP (cf. Sec. 2.1.2) where an optimised policy (dialogue strategy) is learned based on the extended dialogue state (Bui et al., 2009).

For rendering the dialogue adaptive to the user's knowledge or expertise with respect to a given task, Nothdurft et al. (2012) created a dialogue for the task of connecting a Blue-ray player with an amplifier using an HDMI cable. The multi-modal system provides explanations on how to solve the task presenting text, spoken text, or pictures. The system makes assumption over the user knowledge by observing critical events within the dialogue (e.g., failed tries). Based on this knowledge model, the system generates explanations and selects the appropriate type of explanation so that the user can be expected to be capable of solving the task. The knowledge is stored in the knowledge model on a five-step scale where the knowledge fades over time. Earlier, Jokinen and Kanto (2004) also adapted the course of the dialogue to the user's expertise. In a dialogue system providing access to emails for the visually impaired, the system prompts where tailored according to the estimated expertise level of the user. The expertise was derived based on online and offline parameters on three levels distinguishing between *novices*, *intermediate users* and *experts*. Based on this, the amount of help included in regular system utterances was determined.

More than the user's expertise, the user's affective or emotional state is widely considered in dialogue system adaptation. Gnjatović and Rösner (2008) presented work on affective dialogue. For solving the Tower-of-Hanoi puzzle with an SDS, they identify the emotional state of the user in order to recognise if the user is frustrated or discouraged. The dialogue is adapted by answering the questions "When to provide support to the user?", "What kind of support to provide?", and "How to provide support?" depending on the emotional state of the user. By that, the system is capable of providing well adapted support for the user which helps to solve the task.

Aside from task-oriented dialogues, adapting to the affect is very prominent in intelligent tutoring systems (ITS) where the learning path is adapted according to the identified affect (Conati and Maclaren, 2005; Klein et al., 2002; Litman and Forbes-Riley, 2014; Forbes-Riley and Litman, 2011, 2012).

Within this thesis, a major emphasis is laid on adaptivity to user satisfaction or, in a wider context, aspects which may be attributed to events occurring during the interaction. Here, a basic mechanism is provided by VoiceXML (VXML) (Oshry et al., 2007). VoiceXML is a definition of a mark-up language based on XML with the purpose of creating voice applications. VXML defines a set of tags which may then be interpreted by a voice browser in order to provide access to the speech front-end (speech recognition and speech synthesis). By interpreting the user input, different paths may be taken (by loading different VXML documents) for fulfilling the dialogue goal. Adaptivity to interaction-related events may by introduced by, e.g., counting the number of ASR performance events like the number of "nomatches" or "noinputs". Based on these counts, a suitable strategy may be selected.

Similar to this mechanism, prominent work has been presented by Litman and Pan (2002). They identify problematic situations in dialogues by analysing the performance of the speech recogniser (ASR) and use this information to adapt the dialogue strategy. Each dialogue starts off with an user initiated strategy without confirmations. Depending on the ASR performance, may eventually employ a system-directed strategy with explicit confirmations. Applied to TOOT, a system for getting information about train schedules, they achieved significant improvement in task success compared to a non-adaptive system.

Following a slightly different approach, the dialogue initiative is adapted within the MIMIC (Mixed-Initiative Movie Information Consultant) system used for information queries about movies and movie theatres (Chu-Carroll, 2000). To select the type of initiative of the system action, cues from the dialogue interaction are extracted These cues are related to the discourse, e.g., `TakeOverTask` which may be identified by interpreting the semantic input together with the dialogue history. A second category are analytical cues, e.g., `InvalidAction`, which require domain knowledge. Based on the detected cues, appertaining probability distributions are updated which are then used to infer the type of initiative of the next system action thus allowing for multiple initiative shifts during the ongoing interaction.

Litman and Pan (2002), Chu-Carroll (2000) and the mechanisms provided by VXML use interaction related parameters like the ASR performance or discourse cues for adapting the dialogue flow. In our work, we go one step beyond by using the user satisfaction for dialogue adaptation. While our notion of user satisfaction—the Interaction Quality—is also based on interaction related parameters, this parameter-satisfaction relationship is quite complex and may not easily be modelled using rules (Ultes and Minker, 2013c). Hence, the resulting adaptation rules used by Litman and Pan (2002), Chu-Carroll (2000), and VXML merely result in a dialogue strategy which aims at improving the user satisfaction.

While all work described so far on extending the dialogue state models the dialogue management as some variant of a rule-based model, Bui et al. (2007, 2009) proposed an extension of a statistical dialogue manager with an affective state. They extended the state of a POMDP with a component modelling the stress-level of the user to enable the system to select strategies based on the stress of the user. In contrast to other work on POMDP for dialogue management (cf. Sec. 2.1.2), Bui et al. introduced Dynamic Decision Networks to make belief-update tractable.

Evaluation on route navigation in an unsafe tunnel showed that with rising stress level this approach outperforms three simple handcrafted strategies and one random strategy.

Unfortunately, Bui et al. (2009) only used evaluated their approach in a setting where the user state, i.e., the stress level, was simulated. Having a module actually deriving the user state during the dialogue interaction may lead to different behaviour, as the recognition of the user state may be error-prone. Furthermore, while the stress level is a worthwhile user state for certain situations, it is not as general as, e.g., the user satisfaction. In our work, though, we will introduce a real estimation module for adapting the dialogue to user satisfaction.

Naturally, not only the dialogue state may be extended with the user state to achieve user-adaptive dialogue management. For statistical dialogue management like employing a POMDP, reinforcement learning is usually applied to find an optimal strategy automatically. To achieve this, reinforcement learning is based on a manually defined reward function indicating which dialogues are considered good or bad examples. Here, the user state may also be considered as has been shown by the work presented in the following.

3.2.2 Reward Modelling

In reinforcement learning approaches for dialogue management, an optimal policy has to be found automatically. For this, a reward function is used for rewarding good dialogues and penalising bad ones. While most dialogue systems rely on a handcrafted reward function (e.g., a small negative reward per regular turn and in the end a high positive reward for successful and a high negative reward for failing dialogues), many researchers work on finding a suitable reward function automatically. Here, the idea of inverse reinforcement learning (Russell, 1998) has been applied for dialogue management with the goal to learn a reward function from human-human dialogues (Paek and Pieraccini, 2008) or simulated dialogues in a Wizard-of-Oz setting (Boularias et al., 2010).

In this work, though, we are interested in using the user state for reward modelling. Naturally, not all types of user state are equally applicable. User satisfaction, which we will focus on, represents a plausible option. Others have also presented work on using a user satisfaction score to model the reward function.

Here, beginning with Walker et al. (1998a), many teams have employed a reward function based on the idea of the PARADISE framework (Walker et al., 1997). Walker et al. (1998a) and Walker (2000) applied RL to the MDP-based dialogue system ELVIS for accessing emails over the phone. By modelling the reward function using PARADISE, they were able to show that the resulting policy improved the system performance in terms of user satisfaction significantly. The resulting best policy indicated, among other aspects, that the system-initiative strategy was found to work best.

Rieser and Lemon (2008b,a) picked up the idea of using PARADISE for modelling the reward function. Within their multimodal in-car interface to a music database, they used reinforcement learning to automatically identify the best strategy to present query results to the user (speech or screen). For modelling the reward function, they concentrated on a subset of PARADISE labelled as task ease. The authors created a linear regression model to automatically determine the task ease of the current dialogue taking into account the task completion, dialogue length and a multimodal score (i.e., how the users liked the selected presentation variant of the results). Rieser and Lemon proofed their hypothesis that the learned strategy outperforms a handcrafted one with simulated and real users.

While both approaches use real users at some point to create the linear regression model, El Asri et al. (2013, 2012) used expert raters instead to instantiate the PARADIS-like reward function. They presented work on learning a local reward function based on performance scores for complete dialogues (El Asri et al., 2012). For this, they use *reward shaping* and *distance minimisation* for modelling the reward. The performance score ranging between -1 and $+1$ have been annotated using questionnaires. During the training, these scores have been automatically determined using a linear regression model. They compared the two reward modelling methods with only using the scores as reward function (El Asri et al., 2013) thus also comparing dialogue-level reward modelling with turn-level modelling. They showed that applying reward shaping works best in their setting.

Others have also used user satisfaction scores for dialogue-level reward modelling. Meguro et al. (2010), for example, modelled the reward based on user satisfaction annotated by expert raters in a listening-oriented dialogue system. For task-oriented dialogue, Gačić et al. (2013a) proposed a reward function model also based on user ratings on the dialogue level. For their POMDP-based dialogue manager in the restaurant information domain, Gašić et al. used ratings which are have been acquired using Amazon Mechanical Turk. They show that their approach converges much faster than conventional approaches, e.g., using a user simulator.

For modelling the reward incorporating some variant of user satisfaction, all presented work rely on ratings—either from users or experts—which have been applied for one complete dialogue. In our work, though, we will use the Interaction Quality which provides user satisfaction ratings for each exchange. This will allow us for the first time to introduce user satisfaction into the reward function for each single exchange.

3.3 Conclusion on Related Work

In this chapter we have presented related work on modelling the dialogue adaptive to the user state. This includes both work on automatically recognising the user states as well as incorporating the user state into the dialogue manager.

For user state recognition, we have presented work on recognising the four user states *user satisfaction*, *perceived coherence*, *emotion*, and *intoxication*. To model user satisfaction, we have identified the Interaction Quality as the best performing metric as it relates events of the interaction with the satisfaction of the user. While already satisfying performance has been achieved for automatically recognising Interaction Quality, previous work does not take into account the temporal dependencies inherent in dialogue interaction. Hence, we will contribute to the state of the art by performing a deep analysis of these temporal dependencies. This includes a novel approach for Interaction Quality recognition based on Markov models and error correction.

While there is only limited work on coherence recognition on the exchange level, some have already tackled the problem using dialogue act sequences. However, we belief that the coherence should additionally be related to events of the interaction. Hence, we will present novel work on coherence recognition taking into account interaction parameters.

Automatically recognising emotions is a well investigated field within the speech community. However, while several classification approaches and feature combinations have been investigated, a personalisation of the recognition process has not been considered yet. Therefore, we will be the first to present an approach successfully including speaker information into the recognition process.

Intoxication recognition is a very specialised research area which has mostly been considered within a special challenge. Again, many different approaches have been successfully applied. Here, we will contribute to the state of the art by comparing the intoxication recognition performance of machines to humans.

In the field of user-adaptive dialogue management, we have identified two major fields of adaptivity: by extending the dialogue state and by modifying the reward function. While many have adapted the dialogue by extending the state, e.g., with emotions, expertise, or stress, others have taken into account parameters of the interaction, e.g., the performance of the speech recogniser. However, we will be the first to extend the user state with user satisfaction directly. For reward modelling, though, many have modelled the reward taking into account some means of user satisfaction. However, their satisfaction was only determined at the end of the dialogue. Here, we will contribute to the state of the art by using a notion of user satisfaction which allows to incorporate the user satisfaction into the reward at the exchange level.

Furthermore, unlike all presented work on user-adaptive dialogue management which uses the user state as a basis for selecting the next system action, we will present novel work on predicting the influence of the selected system action on the user state and selecting the system action accordingly. We will present an example for this taking into account the perceived coherence.

Summarising the above statements, all modules where we will introduce novelty within this thesis is depicted in Figure 3.2. Here, introducing user state recognition into the dialogue system will be one major part of our contribution (Chapter 4). This includes proposing new concepts for creating the recognition module for the user states *user satisfaction*, *perceived coherence*, *emotion*, and *intoxication*. Some of the concepts will require a full implementation of the model. Furthermore, we will contribute to the state-of-the-art by providing two corpora with annotated data for *user satisfaction* and *perceived coherence*.

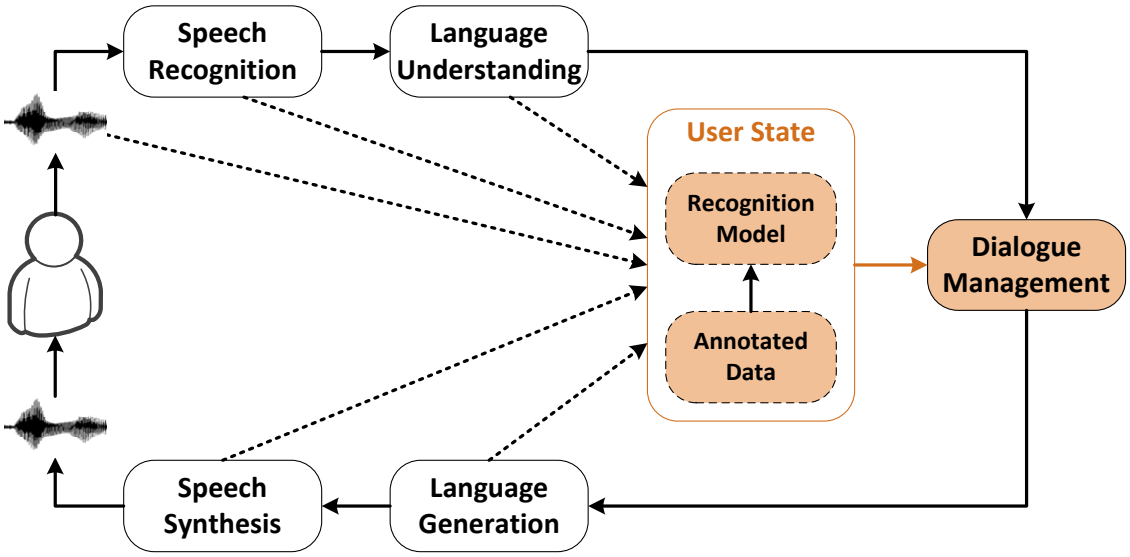


Fig. 3.2: The novelty of our work. All parts of the SDS addressed by the novelty of this thesis are displayed on the right hand side including the User State and the Dialogue Management.

The second major part of the novelty contained in this thesis lies within the dialogue management (Chapter 5). We will propose concepts for incorporating the user state into the dialogue management process using rule-based as well as fully statistical approaches. Again, these concepts also required a complete redesign and implementation of the dialogue manager itself.

With the general goal user-centred adaptive spoken dialogue modelling where the course of the dialogue is adapted based on the recognised user state, we will present our own work on user-adaptive dialogue management in Chapter 4. First, though, we will continue in the next chapter with our work on user state recognition.

4

User State Recognition

Proposing novel approaches for automatically recognizing the user state represents the first major contribution of this work. With the ultimate goal of rendering communication between humans and machines to be human-like, the interaction needs to be more user-centered. To achieve this, the system should not only understand what the user is saying. On top of this, the system should be aware of the user state, e.g., the user's emotions, intoxication level, or satisfaction level. In this chapter, we will propose new concepts and methods for recognizing four different types of user state, which are *User Satisfaction*, *Perceived Coherence of System Actions*, *Emotion*, and *Intoxication*.

To have those user states as input to adaptive dialogue management, they should meet certain criteria which have been outlined for Interaction Quality (Ultes et al., 2012c) but most translate to all user states:

- exchange-level user state recognition,
- automatically derivable features,
- domain-independent features,
- consistent labelling process,
- reproducible labels, and
- unbiased labels.

As dialogue management is performed after each system-user exchange, dynamic adaption of the dialogue strategy to the user state requires exchange-level user state recognition. Therefore, approaches which derive a user state over a complete dialogue are of no use.

Features serving as input variables for a statistical classification model need to be automatically derivable from the dialogue system modules. This is important because semi-automatic or manually derivable features, e.g., manually annotated dialogue acts, produce high costs and are also not available immediately during run-time in order to use them as additional input to the dialogue manager. Furthermore, for creating *general* user state recognition modules, the features should be domain-independent, i.e., not depending on the task domain of the dialogue system.

Another important issue is the consistency of the labels. Labels applied by the users themselves are subject to large fluctuations among the different users (Lindgaard and Dudek, 2003). As this results in inconsistent labels, which do not suffice for creating a generally valid model for the user state, ratings created in controlled conditions yield more consistency. One example would be to use expert raters which are asked to annotate the respective user state following previously established annotation guidelines. This also minimises the risk of being influenced by

certain aspects of the SDS which are not of interest in the respective context, e.g., the character of the synthesised voice. Therefore, expert raters create less biased labels.

The four user states which are regarded within this chapter all suffice these criteria. Therefore, we will propose novel approaches for recognising these user states using supervised learning techniques and statistical classification methods and investigate the viability of those approaches with extensive evaluation and performance tests in various configurations. Moreover, we will present two annotated corpora which have been created within the scope of this thesis. Both corpora will be used for evaluating our approaches on recognising the respective user state.

The remainder of the chapter is grouped by user state. Hence, each section starts with a descriptions of the respective user state followed by our contribution to the state of the art: for each user state, we will propose novel recognition approaches, describe the implementation of the used statistical models, and provide a thorough evaluation. An emphasis is placed on user satisfaction recognition in the first section describing several novel approaches—both static and sequential—with the main focus on analysing and exploiting the temporal nature of user satisfaction recognition. The other user states are described in the subsequent sections in the order of their importance for adaptive dialogue modelling: perceived coherence, emotions, and intoxication.

4.1 User Satisfaction Recognition in Spoken Dialogue Systems

One condition which may occur in almost all types of human-system interaction is whether the user of the system is satisfied or not. This is even more prominent in task-oriented applications where the interaction has a well-defined goal: to get the task done. If the system behaves in an uncooperative manner, the user might get frustrated and will most likely not be satisfied with the interaction. One possible way of dealing with these situations is to automatically detect the satisfaction level of the user in order to use this information to adapt the course of the dialogue. To do so, machine learning methods may be applied. In this section, we will propose several novel classification approaches to automatically predict the user satisfaction level.

However, first, a better understanding of the term *user satisfaction* is necessary. Unfortunately, there is no rigorous definition of the term "user satisfaction" in the literature as already addressed by Schmitt and Ultes (2015). Doll and Torkzadeh (1991) describe *user satisfaction* as the opinion of the user about a specific computer application. Other terms for *user satisfaction* are common, e.g., "user information satisfaction", which is defined as "the extent to which users believe the information system available to them meets their information requirements" (Ives et al., 1983). For Larcker and Lessig (1980) address *user satisfaction* as "perceived usefulness of information". User satisfaction and usability are closely interwoven. (ISO, 1998) defines "usability" by subsuming a compound of efficiency, effectiveness and satisfaction. Yet *user satisfaction* is often seen as a by-product of great usability in the HCI literature (Lindgaard and Dudek, 2003).

While the definition of *user satisfaction* represents a completely subjective metric based on user judgements, the term *quality* includes both, subjective and objective characteristics: Möller et al. (2009) presented a taxonomy of quality criteria related to HCI describing quality as a bipartite issue consisting of *Quality of Service (QoS)* and *Quality of Experience (QoE)*. QoS is defined as "the collective effect of service performance, which determines the degree of satisfaction of the user" (ITU, 1994) and depicts an objective view on quality. Thus, it can be determined by the

system developer. In contrast to this stands QoE, which is defined as "the overall acceptability of an application or service, as perceived subjectively by the end-user." (ITU, 2007). This definition implies that QoE requires a subjective judgement process by the user and can thus only be measured with user surveys (Möller et al., 2009). However, measuring QoE, i.e., subjective user judgements about the interaction, turns out to be difficult, for three reasons:

Strong variations
: Different users have a different understanding (and thus expectation) of a functioning interaction. Some might perceive a dialogue behaviour, e.g., a perpetual posing of the same question, as very disturbing whereas others would not even take notice of it. Hence, this implies that user satisfaction is user-specific and requires models trained on data from a specific user (cf. Engelbrecht et al., 2009).

Biased judgements
: Users judging the interaction with the system may be biased by unrelated events. Here, Engelbrecht and Möller (2010) have shown that people's emotions influence their perception of user satisfaction. The user having a hard day at work, for instance, might reflect on the judgement of the interaction thus not generating a fair rating.

Data collection problem
: Tracking real user satisfaction online, i.e., throughout a spoken interaction, with the aim to create user-specific models is only possible under laboratory conditions. The users will then pursue a fake task in an artificial environment. It can hardly be examined to which extent this scenario differs from a real-life scenario with disturbing environmental factors and a real user need. It should be further noted that a user will have to synchronously interact with a system and adjusting his satisfaction at the same time. However, this depicts a high cognitive load for the test subject and in the worst case could falsify the user's judgement about the interaction.

To overcome these problems, employing expert raters to judge the interaction proposes one solution. Having a third person observing the interaction allows for using data of real, live systems thus solving the data collection problem. Furthermore, strong variations within the judgements are evened out and the experts are less likely to be biased by unrelated events. Here we have shown that this also results in a less cost-intensive annotation process (Ultes et al., 2013b).

A metric representing user satisfaction and at the same time accounting for the before mentioned problems is defined within the Interaction Quality paradigm described and analysed in detail by Schmitt and Ultes (2015) and originally presented by Schmitt et al. (2011a). Hence, before we continue with our own work on user satisfaction recognition, we will briefly describe the work of Schmitt et al. by presenting the Interaction Quality paradigm.

4.1.1 Interaction Quality Paradigm

Interaction Quality (IQ) has been proposed by Schmitt et al. (2011a) as an alternative and more objective measure of user satisfaction. It overcomes the problems stated above by introducing expert raters instead of using real users for assigning the satisfaction label. Here, we were able to show that ratings applied by experts are suitable for substituting user ratings and that both have a high correlation (Ultes et al., 2013b).

The Interaction Quality paradigm describes a general scheme shown in Figure 4.1 including not only the annotation of IQ but also the creation process of a statistical model for recognising IQ for each system-user exchange within human-machine dialogue interaction.

IQ itself is defined for each system-user-exchange with values on a scale from five to one: 5 ("satisfied"), 4 ("slightly unsatisfied"), 3 ("unsatisfied"), 2 ("very unsatisfied"), and 1 ("extremely unsatisfied"). Each dialogue starts with the rating "satisfied" as the metric only models whether problems occur within a dialogue. In the beginning of the dialogue, nothing unsatisfying has happened yet though. An example for an annotated dialogue is presented in Figure 4.2.

To automatically estimate IQ, the paradigm describes beside the annotation of the training dialogue with IQ the automatic extraction of parameters describing the current status of the interaction. These parameters are then used as input variables to a statistical classification model which predicts the IQ level of the ongoing interaction at the current system-user-exchange.

The input variables called interaction parameters are rendered on three levels as shown in Figure 4.3: the exchange level, the window level, and the dialogue level. The exchange level comprises parameters derived from SDS modules Automatic Speech Recogniser, Spoken Language Understanding, and Dialogue Management directly, e.g., whether the user input was understood or rejected by the speech recogniser. Parameters on the window and the dialogue level are sums, means, frequencies or counts of exchange level parameters, e.g., the count of how often the user input was rejected by the speech recogniser. While dialogue level parameters are computed out of all exchanges of the dialogue up to the current exchange, window level parameters are only computed out of the last *m* exchanges.

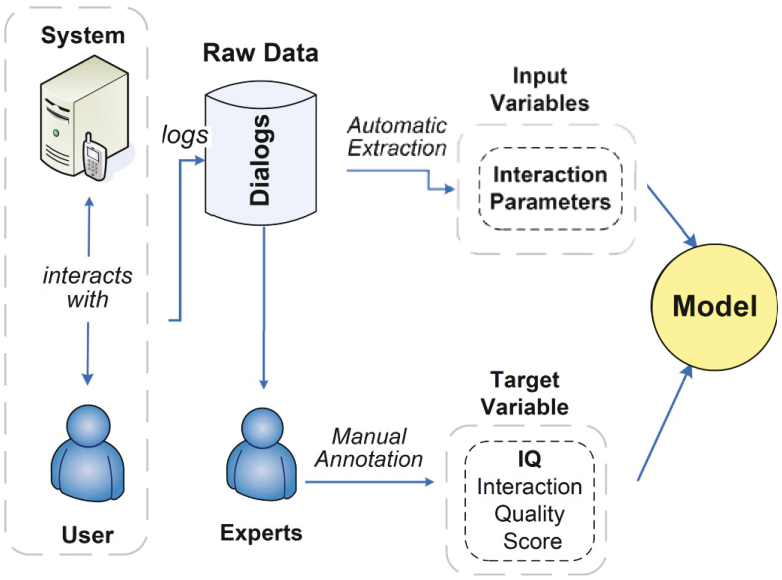


Fig. 4.1: The Interaction Quality scheme originally published by Schmitt and Minker (2013) describing the overall process. Dialogues between the user and the system are logged and annotated with IQ by expert raters. Interaction Parameters are automatically extracted from the logs and used as input variables to train a statistical model targeting IQ.

		Utterance	
(1)	S:	Welcome to the CMU Let's Go bus information system.	
(2)	U:		$IQ = 5$
(3)	S:	To get help at any time, just say Help or press zero.	
(4)	U:		$IQ = 5$
(5)	S:	What can I do for you?	
(6)	U:	PENN AT BUTLER TIME IS ...	$IQ = 5$
(7)	S:	Leaving from OAKLAND . Is this correct?	
(8)	U:	OAKLAND	$IQ = 4$
(9)	S:	Leaving from OAKLAND . Is this correct?	
(10)	U:	YES	$IQ = 4$
(11)	S:	Right	
(12)	U:	—	$IQ = 4$
(13)	S:	Where do you want to go?	
(14)	U:	SQUIRREL HILL OF FIFTH AVENUE	$IQ = 4$
(15)	S:	Going to FIFTH AVENUE . Is this correct?	
(16)	U:	YES	$IQ = 5$
(17)	S:	Alright	
(18)	U:	—	$IQ = 5$
(19)	S:	When do you wanna travel?	
(20)	U:	SUNDAY	$IQ = 4$
(21)	S:	At what time do you wanna travel?	
(22)	U:	AT FIVE FOUR VERY ABOUT ELEVEN O'CLOCK	$IQ = 4$
(23)	S:	Leaving at $IQ = 11$ a.m. . Is this correct?	
(24)	U:	YES	$IQ = 5$
(25)	S:	Right	
(26)	U:	—	$IQ = 5$
(27)	S:	Hold on. Let me check that for you.	
(28)	U:	—	$IQ = 5$

Fig. 4.2: Excerpt from a dialogue annotated with Interaction Quality for each system-user-exchange. (CallID 2061123000 of the LEGO corpus.)

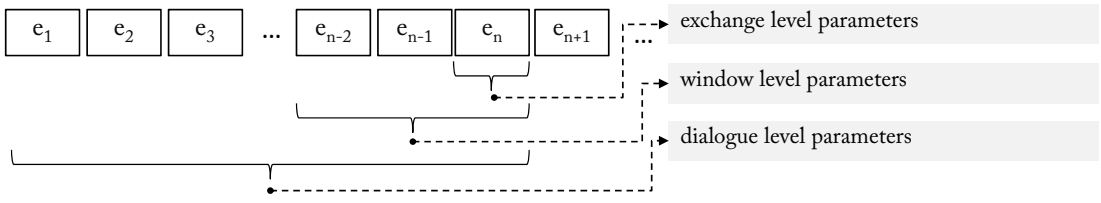


Fig. 4.3: The interaction parameters consist of three levels originally published in (Schmitt et al., 2012): the exchange level containing information about the current exchange, the window level, containing information about the last m exchanges (in this example, $m = 3$), and the dialogue level containing information about the complete dialogue up to the current exchange.

A list of all automatically derived features along with a brief description may be found in Table B.1. For a more elaborated description, refer to Schmitt and Ultes (2015).

As we have presented the general notion of Interaction Quality by Schmitt et al. (2011a), the remaining section describes solely or own original work on Interaction Quality recognition and related aspects. Consequently, by applying the Interaction Quality paradigm, two corpora have been created. Their annotation process is described in detail in Section 4.1.5. These corpora are also used to evaluate the different approaches for estimating IQ. These approaches—representing our contribution to the state of the art in Interaction Quality recognition—are described in detail in the following.

4.1.2 Static Methods for Interaction Quality Recognition

For automatic estimation of Interaction Quality, we have applied several approaches rendering the problem as a supervised classification task. In this work, we distinguish between *static* and *sequential* approaches. Having a structure where each system-user-exchange is part of a sequence of exchanges, i.e., the dialogue, the act of estimating IQ for each exchange may be viewed as a *static* task by regarding each exchange independently of all other exchanges. Thus, the exchange is not embedded in the dialogue it belongs to. Having the exchange being embedded, though, results in a *sequential* approach which is based on the fact that the value of the current exchange highly depends on the value of the previous exchange. In this section, though, we will only look at the static approaches having the sequential approaches being described in Section 4.1.3.

Standard Machine Learning Methods

The initial approach by Schmitt et al. (2011a) was to apply a Support Vector Machine (SVM, see Sec. 2.2.1) to estimate IQ. Hence, we will also apply a SVM with the general framework being depicted in Figure 4.4: having a training set of m vectors f containing n interaction parameters each, the SVM is trained in a supervised learning setting using the IQ references r_i corresponding to each vector. The resulting SVM model is then used in the estimation phase to estimate a hypothesis h_{IQ} for IQ based on one sample of interaction parameters.

SVMs are based on linear discrimination: to find a hyper plane in space which discriminates two classes with a maximum margin. While many classifiers use this concept, we are also interested in how classifiers with other characteristics perform. Hence, we also apply a Naïve Bayes method (see Sec. 2.2.1) which models the problem using probability distributions. Finally, a rule learner (see Sec. 2.2.1) is applied deriving a set of rules to cover the classification problem. Both approaches follow the same pattern of SVM classification depicted in Figure 4.4: in the training phase, samples with IQ references are used to train the model. The trained model is then used in the evaluation phase to estimate the IQ hypothesis for a given interaction parameter vector.

These static standard machine learning techniques form the basis for more elaborated recognition methods like the Hybrid Hidden Markov Model (Sec. 4.1.3) and a Hierarchical Error Correction approach described in the following. The performance evaluation of these standard approaches will be presented in Section 4.1.6.

Hierarchical Error Correction

Applying statistical classification algorithms for estimating IQ will always result in less than 100% accuracy. In fact, almost all classifiers applied for almost any problem will not always

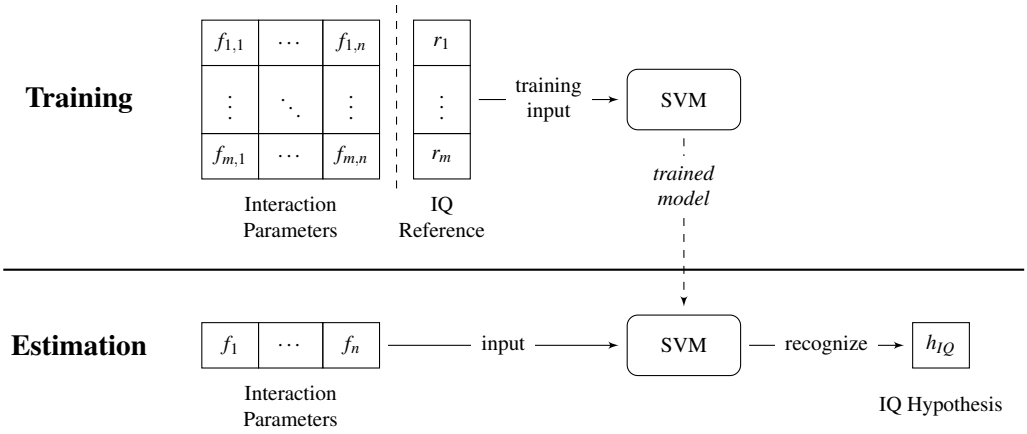


Fig. 4.4: To create a statistical model for IQ recognition (here an SVM), a set of m interaction parameter vectors $f_{m,n}$—consisting of n interaction parameters each—along with their annotated IQ references r_m are used. To estimate the hypothesis h_{IQ}, the trained model is used to evaluate one input sample.

estimate the correct class. Hence, we propose to estimate the error the classifier makes when estimating a hypothesis and correct this hypothesis by the estimated error (Ultes and Minker, 2013b). Furthermore, the proposed error correction approach is extended by a confidence model which allows for several different ways of deriving the final hypothesis. This plug-in architecture may provide a general approach on improving the estimation performance regardless of the underlying standard machine learning algorithm.

Error correction

To apply error correction for the process of statistical classification, a hierarchical two-stage approach is proposed, depicted in Figure 4.5.

At the first stage, a statistical classification model is created using interaction parameters as input and IQ as target variable. At the second stage, the error e_r of the hypothesis h_0 of the classifier is calculated by

$$e_r = h_0 - r \,, \tag{4.1}$$

where the reference r denotes the true IQ value. In order to limit the number of error classes, the signum function is applied. It is defined as

$$\operatorname{sgn}(x) := \begin{cases} -1 & \text{if } x < 0 \,, \\ 0 & \text{if } x = 0 \,, \\ 1 & \text{if } x > 0 \,. \end{cases} \tag{4.2}$$

Therefore, the error is redefined as

$$e_r = \operatorname{sgn}(h_0 - r) \,. \tag{4.3}$$

Next, a statistical model is created similarly to stage one but targeting the error e_r. The difference is that the input parameter set is extended by the IQ hypothesis h_0 of stage one. For the

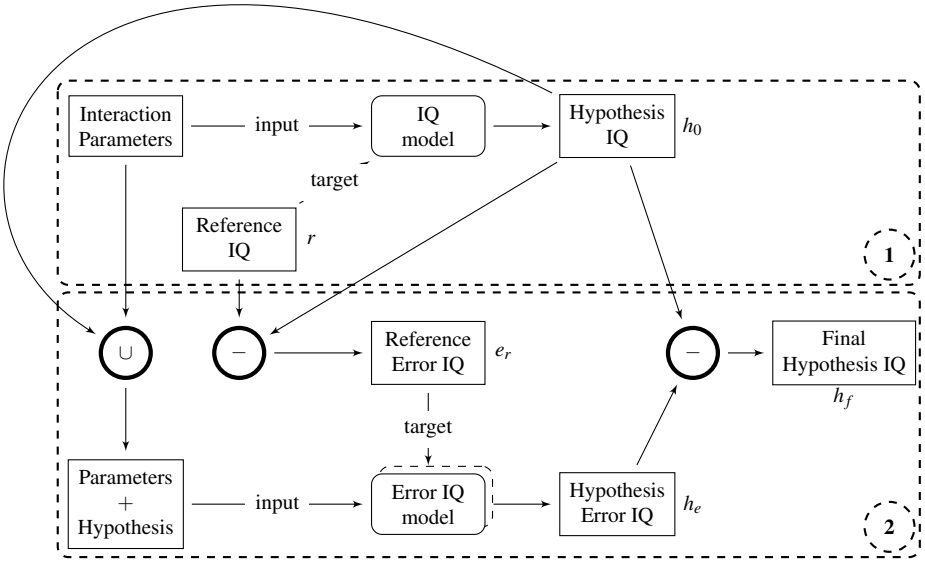


Fig. 4.5: The complete IQ estimation process including error correction originally published in (Ultes and Minker, 2013b). After estimating IQ in Stage 1 (upper frame), the error is estimated and the initial hypothesis is corrected in Stage 2 (lower frame).

model creation, two approaches are applied: creating one model which discriminates between all error classes $(-1,0,1)$ at the same time and creating two models where one estimates the positive $(0,1)$ and one the negative error $(-1,0)$. For the latter variant, the error of the class which is not estimated by the respective model is mapped to 0. By this, the final error hypothesis h_e may be calculated by simple addition of both estimated error values:

$$h_e = h_{e_{-1}} + h_{e_{+1}} \, . \tag{4.4}$$

Combining the hypothesis of the error estimation h_e with the hypothesis of the IQ estimation h_0 at stage one produces the final hypothesis h_f denoting the Interaction Quality estimation corrected by the estimated error of the statistical model:

$$h_f = h_0 - h_e \, . \tag{4.5}$$

As the error estimation will not work perfectly, it might recognise an error where there is none or—even worse—it might recognise an error contrary to the real error, e.g., -1 instead of $+1$. Therefore, the corrected hypothesis might be out of range. To keep h_f within the defined bounds of IQ, a limiting functions is added to the computation of the final hypothesis resulting in

$$h_f = \max(\min(h_0 - h_e), b_u), b_l) \, , \tag{4.6}$$

where b_u denotes the upper bound of the IQ labels and b_l the lower bound.

Confidence Model

To extend the error correction approach with a confidence model, we regard the confidence scores of the static classification methods as probabilities for the respective IQ value[1] and compute the

[1] In general, confidence scores and probabilities not necessarily exchangeable entities.

confidences of the error estimator using probability theory. Hence, we refer to the confidence of h_0, e.g., as $P(h_0)$.

These prior probabilities are directly taken from the confidences of the stage one classifier. Therefore, we need to find a suitable model for the posterior $P(h_f|h_0,h_e)$. For this, the set C_{iq} is defined as

$$C_{iq} := \{(h_0,h_e) : \max(\min(h_0 - h_e), b_u), b_l) = iq\} \tag{4.7}$$

containing all (h_0,h_e) pairs which result in the same final hypothesis h_f^{diff}. For $h_f^{diff} = 5$, $b_u = 5$, and $b_l = 1$, for example, the set would be

$$C_5 = \{(5,+1),(5,0),(4,-1)\} . \tag{4.8}$$

Using this set, the posterior is defined by

$$P(h_f|h_0,h_e) = \begin{cases} P(h_e|h_0)P(h_0) & \text{if } (h_0,h_e) \in C_{h_f} , \\ 0 & \text{otherwise .} \end{cases} \tag{4.9}$$

Here, the confidence of the error classifier $P(h_e|h_0)$ may also be directly taken from the applied error IQ model. For the case of using two classifiers at stage two discriminating between the error classes $(-1,0)$, and $(+1,0)$ separately, the confidence scores of both classifiers are combined using

$$P(h_e|h_0) = P(h_{e_{+1}}|h_0) + P(h_{e_{-1}}|h_0) , \tag{4.10}$$

where $(h_{e_{+1}},h_{e_{-1}}) \in E_{h_e}$. The error set E_e for error class e is similarly defined to C_{iq}:

$$E_e := \{(h_{e_{+1}},h_{e_{-1}}) : h_{e_{-1}} + h_{e_{+1}} = e\} . \tag{4.11}$$

For E_0, for instance, this set would contain two pairs:

$$E_0 := \{(0,0),(+1,-1)\} . \tag{4.12}$$

Finally, the confidences of the final hypotheses for a given IQ value iq are calculated out of the posterior $P(h_f|h_0,h_e)$ by

$$P_f(h_f = iq) := P(h_f) = \sum_{(h_0,h_e)} P(h_f|h_0,h_e) . \tag{4.13}$$

Confidence-based Hypothesis Generation

Based on the confidence measures, multiple ways of generating the final hypothesis h_f exist in addition to just taking the bounded difference in Equation 4.6. The first option is to assign the IQ value with the highest confidence:

$$h_f^{max} = \arg\max_{iq} P_f(iq) . \tag{4.14}$$

This way of deriving the final hypothesis is equal to the way the final hypothesis is derived in regular classification approaches like the applied SVM.

A second way of utilising the confidence measures is to calculate the weighted sum based on the confidence scores:

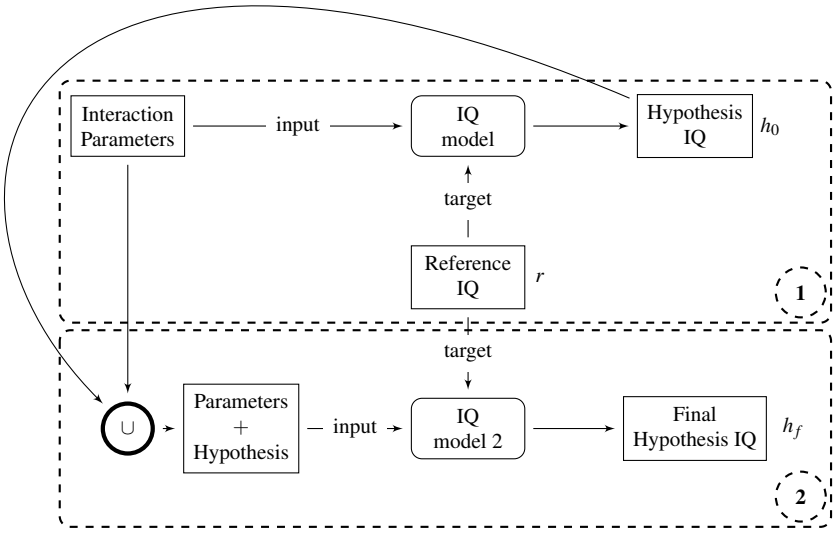


Fig. 4.6: The simple hierarchical model first estimates IQ in Stage 1 classification (upper frame) and then using this hypothesis for a second classification model in Stage 2 (lower frame) also targeting IQ directly.

$$h_f^{wSum} = \text{round}(\sum_{iq} P_f(iq) \cdot iq) \,, \tag{4.15}$$

where round$(\cdot)$ represents the regular rounding function.

To evaluate how the different variants of error correction perform, experiments have been conducted which are presented in Section 4.1.6 and compared to a simple hierarchical approach which will be explained in the following.

Simple Hierarchical Approach

The previously described error correction approach offers means of estimating IQ with a two-stage approach by estimating the error in the second state. In addition, we are interested in whether this two-stage set-up may be used in a simpler way by targeting IQ directly in the second stage (Ultes and Minker, 2013b).

The overall scheme is presented in Figure 4.6. Similar to the error correction approach, the first stage contains the plain application of a statistical method as described in Section 4.1.2. Also, the feature set to the second-stage model is again extended by the hypothesis of the first model. In contrast to error correction, this feature set is then used to directly estimate IQ. Here, the hopes are first that the hypothesis of the first classifier encodes information about the problem which may help the second model to decide. Second, by applying two different types of classifiers, each model might grasp different aspects of the classification problem.

As already stated, modelling the problem of estimating the Interaction Quality for each system-user-exchange in a static way, i.e., regarding each exchange being independent of the dialogue it belongs to, is only one way of tackling the problem of estimating IQ. The problem may also be viewed as a sequential task which will be elaborated on in the next section.

4.1.3 Sequential Methods for Interaction Quality Recognition

While previously presented approaches for estimating the Interaction Quality for each system-user-exchange regard each exchange independently, in fact, it is part of a sequence: the conversation between the system and the user. This dialogue comprises multiple system-user-exchanges in a fixed order and the current exchange's content depends on all previous exchanges. The same is true for the IQ value: the quality of the current exchange depends on how the interaction has evolved until then. Hence, we apply classification models which are able to reflect this property of the Interaction Quality estimation problem. Most prominently, this is the Hidden Markov Model (HMM, see Sec. 2.2.2) which is widely known and applied for such types of problems. Furthermore, a Conditioned Hidden Markov Model (CHMM, see Sec. 2.2.2) and a hybrid version of the regular HMM are considered. In the following, we will describe how those may be applied for IQ estimation.

Hidden Markov Model

When attempting to estimate the IQ value of a sequence of exchanges, only observations o at a certain time can be made. Here, the observations are the interaction parameters derived for each exchange. However, the hidden sequence of IQ values is of interest. Hence, for modelling the IQ estimation using an HMM, the IQ values are regarded to be the hidden random variables (Ultes et al., 2012a). Thus, each hidden state is tied to exactly one IQ value. Within the described work, a model with five states has been chosen. The model is depicted in Figure 4.7. Each hidden state is associated with an own observation probability model reflecting the probability of observing o at the current time t in the connected hidden state s, i.e., having iq being the current quality of the dialogue which is tied to s. The observation probability $b_j(o)$ is modelled using regular Gaussian mixture models (GMMs).

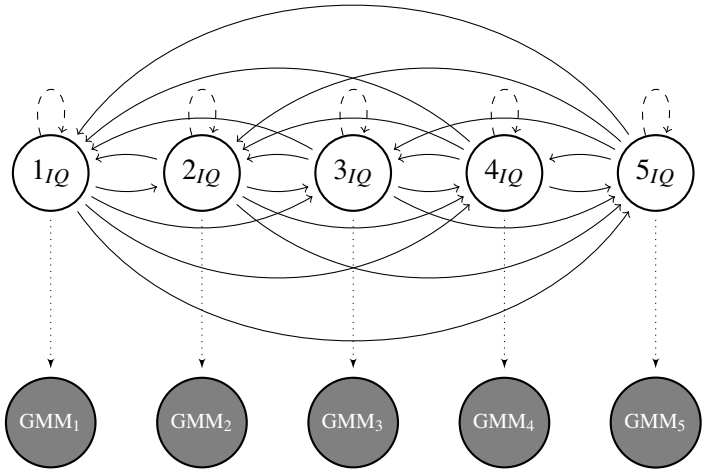


Fig. 4.7: The ergodic model of an HMM used for modelling the sequential IQ estimation problem. For five hidden states, each state is tied to one IQ value and the observation probability is modelled using GMMs.

To evaluate this model the probability $p(q_t|O_t,\lambda)$ of seeing observation sequence $O_t = (o_1,o_2,\ldots,o_t)$ while being in state q_t at time t given the HMM λ is calculated using the Forward Algorithm:

$$\begin{aligned} p(q_t = s_j|O_t,\lambda) &= \alpha_t(j) \\ &= \sum_{i=1}^{|S|} \alpha_{t-1}(i) a_{ij} b_j(o_t) \,. \end{aligned} \tag{4.16}$$

Here, a_{ij} describes the transition probability of transitioning from state s_i to state s_j. To find a suitable model λ, the HMM is trained, for example, by using the Baum-Welch algorithm.

For determining the most likely class $\hat{\omega}_t$ at time t, where each state $j \in S$ is associated with one class ω, the following equation is used:

$$\hat{\omega}_t = \arg\max_j \alpha_t(j) \,. \tag{4.17}$$

In addition to GMMs, there are further options for modelling the observation probability. One interesting option is to use confidence scores of static classifiers resulting in a hybrid approach which will be described next. The evaluation of the HMM will be presented in Section 4.1.6.

Hybrid Hidden Markov Model

While the observation probability of a Hidden Markov Model is classically modelled with GMMs, we are also proposing another approach combining static classification approaches with HMMs resulting in something which we call Hybrid Hidden Markov Model (Ultes and Minker, 2014a). To take advantage of existing static classification approaches, the Hybrid-HMM is modelled as a plug-in structure where virtually any static classification algorithm may be used.

For applying an HMM while exploiting existing statistical classification approaches, the observation function $b_j(o_t)$ is modelled by using confidence score distributions of statistical classifiers, e.g., a Support Vector Machine. Furthermore, the transition function a_{ij} is computed by taking the frequencies of the state transitions contained in the given corpus. Therefore, an ergodic HMM is used comprising five states with each representing one of the five IQ scores. The model is depicted in Figure 4.8.

Moreover, in SDSs, a system action act is performed at the end of each system turn. This can be utilised by adding an additional dependency on this action to the state transition function a_{ij}. By augmenting Equation 4.16, this results in

$$\alpha_t(j) = \sum_{i=1}^{|S|} \alpha_{t-1}(i) a_{ij,\mathsf{act}} b_j(o_t) \,. \tag{4.18}$$

This refinement models differences in state transitions evoked by different system actions, e.g., a different transition probability is expected if a WAIT action is performed compared to a CONFIRMATION. Therefore, two versions of the Hybrid-HMM are evaluated: an action-independent version as in Equation 4.16 and an action-dependent version as in Equation 4.18[2].

[2] Equation 4.18 is equal to the belief update equation known from the Partially Observable Markov Decision Process formalism (Kaelbling et al., 1998).

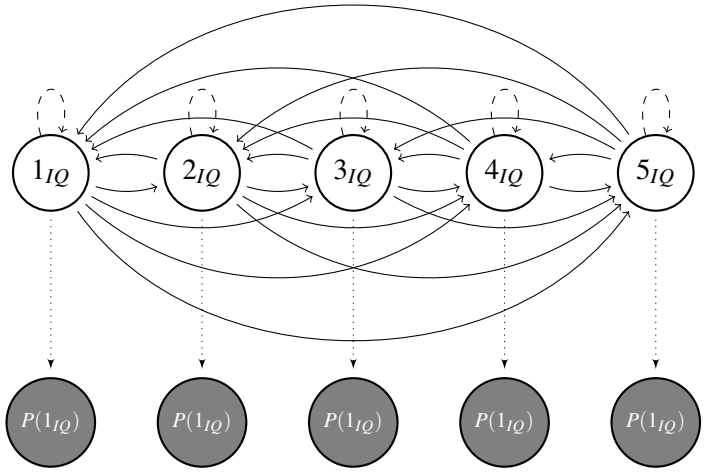


Fig. 4.8: The ergodic model for a Hybrid-HMM used for modelling the sequential IQ estimation problem. As for the classical HMM, all five hidden states are each tied to one IQ value. However, in contrast to the HMM, the observation probability is modelled by confidence scores of static classifiers.

Finally, computing the frequencies of the state transitions contained in the given corpus for modelling the transition probability may not be adequate as this highly depends on the section of the data which is available. However, for having action independent transitions, the transition function is only a $5x5$ matrix which can easily be defined manually. Hence, we will also analyse the performance of the Hybrid HMM using handcrafted transition matrices. To reflect the rating guidelines which do not allow for transitions to IQ values being non-neighbours, the model is further restricted as shown in Figure 4.9.

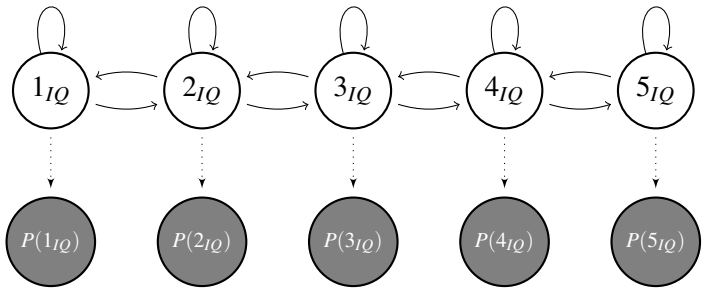


Fig. 4.9: The Hybrid HMM with a restricted transition function allowing only transitions between neighboring IQ values. This restriction is applied for the case of modelling the transition probability manually. Being a *Hybrid* HMM, the observation probability is modelled by using confidence measures of static classifiers.

As the two approaches based on the classical HMM only yield a class probability indirectly by tying the IQ values to the hidden states, the Conditioned HMM is also employed for IQ estimation

as it combines the advantage of the HMM inherently modelling sequential data while providing a class probability directly. Hence, the Conditioned HMM will be described in the following while the performance of the Hybrid-HMM may be found in Section 4.1.6.

Conditioned Hidden Markov Model

Classical Hidden Markov Models estimate a probability $p(\circ|\lambda)$of observing the observation sequence $\circ$ given the model parameters λ. Unfortunately, there is no straight-forward way of using an HMM for classification tasks where each observation is related to one class. In our task of estimating IQ, for example, this is resolved by tying each state to one of the classes and determining the most likely state sequence[3]. The Conditioned Hidden Markov Model (CHMM) compensates for that by introducing a new set of nodes which represent the classes thus allowing for a direct class estimation. The general definition of a CHMM is elaborated in more detail in Section 2.2.2.

The model of a CHMM for estimating IQ is depicted in Figure 4.10 as originally applied by Ultes et al. (2012a). Similar to the case of a classical HMM, an ergodic model has been chosen which allows transitions from each hidden state s_i to any other hidden state s_j. However, an arbitrary number of hidden states is possible without a clear mapping to one of the target IQ values. This is as IQ is represented by the class states. Hence, a class probability may be computed directly for each part of the sequence. Again, Gaussian mixture models are used to estimate the observation probabilities. The CHMM will be evaluated in Section 4.1.6.

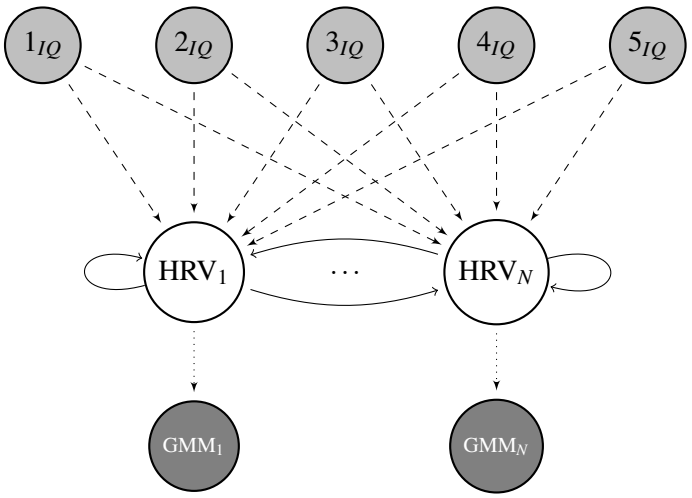


Fig. 4.10: The ergodic model used for modelling the sequential IQ estimation problem with an CHMM. For five label states, where each state is tied to one IQ value, an arbitrary number of hidden states with corresponding GMM modelling the observation probability may be used.

While we have thoroughly described our proposed novel methods for estimating the Interaction Quality, we will continue with giving implementation details for all proposed statistical models used within our concepts for IQ recognition.

[3] Acoustic modelling for speech recognition using HMMs is also done this way, cf. 2.1

4.1.4 Model Implementation for Interaction Quality Recognition

To evaluate the presented static and sequential approaches, mostly, existing implementations and libraries have been used. All standard static machine learning methods (see Sec. 2.2) are part of the analytic software RapidMiner[4] which provides implementations for a wide range of statistical models including Naïve Bayes, the SVM library libSVM (Chang and Lin, 2011), the SVM implementation based on Sequential Minimal Optimisation (Platt, 1999), as well as a rule learner based on RIPPER (Cohen, 1995). These classifiers were also used for the error correction approach.

For the evaluation of the Hidden Markov Model approach, we relied on the publicly available JaHMMlibrary (Francois, 2006), an implementation of a Hidden Markov Model in Java. This was not necessary for the Hybrid-HMM where the models are not trained. The transition model is based on corpus frequencies or handcrafted and the observation model utilises confidence measures of static standard machine learning algorithms again implemented by RapidMiner. The Forward-Backward algorithm used for calculating the probability distribution over all hidden states was implemented within a Perl script.

Evaluating the Conditioned Markov Model approach was more challenging as there was no library implementation yet. Hence, we created our own library JaCHMM (Ultes et al., 2013a) based on JaHMM which we will give more details about in the following.

The JaCHMM[5] implements a version of the CHMM equations (see Sec. 2.2.2) which introduced an additional independence assumption having the transition probabilities being independent of the class label. JaCHMM provides the following features:

Labels	JaCHMM may be used for data where a label is related to a whole sequence of observations as well as where a label is related to one single observation.
Observations	Observations may either be discrete or continuous. Therefore, different types of observation probability distributions have been implemented, i.e., discrete probabilities and Gaussian mixture models for continuous observations.
Initialisation and Training	Initialisation is implemented using the k-means algorithm, which can also be used for training. Additionally, training can be performed by using the traditional Baum-Welch algorithm.
Computational Efficiency	In order to increase the computational efficiency of JaCHMM, reasonable independence assumptions regarding the transition probability $a_{ij,y} = p(\mathrm{w}^{(t)} = w_j \vert \mathrm{w}^{(t-1)} = w_i, \mathrm{y}^{(t)} = y)$ have been introduced, resulting in the simplified version $a_{ij,y} = p(\mathrm{w}^{(t)} = w_j \vert \mathrm{w}^{(t-1)} = w_i) \cdot p(\mathrm{w}^{(t)} = w_j \vert \mathrm{y}^{(t)} = y)$. Within the JaCHMM library, both variants are implemented.
Algorithms	For their application in the Conditioned Hidden Markov Model, prominent algorithms known from HMMs were adapted, e.g., the Viterbi, Forward-Backward, Baum-Welch, or K-Means algorithm.

The command-line interface offers commands for creating, initialising, learning, and evaluating CHMMs enabling the library to work as stand-alone software.

[4] `http://www.rapidminer.com`

[5] The JaCHMM library is available online under the BSD license at `http://nt.uni-ulm.de/ds-jachmm`.

Like the JaHMM, the JaCHMM is designed to achieve reasonable performance without making the code unreadable. Consequently, it offers a simple way of applying the Conditioned Hidden Markov Model in various tasks, e.g., for scientific or teaching purposes.

To generate performance measures in a proof-of-concept application of the JaCHMM library, a three-class problem was chosen. Figure 4.11 shows the average time needed per observation sequence in *ms* for training and evaluating using the JaCHMM and comparing it to JaHMM with respect to the number of hidden states. While training of the HMMs (having one HMM per class) was much faster for all configurations, for a small number of hidden states, evaluation of the CHMM was faster than evaluating three HMMs. It should be noted, that for HMM performance, the total number of hidden states as well as the training and evaluation time was summed up over all HMMs.

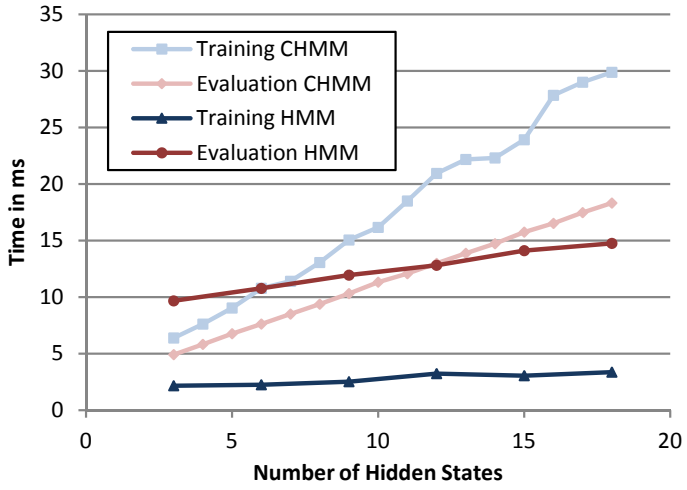


Fig. 4.11: Average time needed for training and evaluation of the CHMM and HMM per observation sequence originally published in (Ultes et al., 2013a). For the HMM, the total number of hidden states and the time for training and evaluating summed up over all HMMs is used.

While we have already proposed a number of different static and sequential classification approaches, the question remains how they perform. Hence, all approaches are evaluated thoroughly and compared against each other. However, for this, a defined data set is needed. Hence, two corpora have been created which will be described in the following section along with a detailed description of how the ground truth, i.e., the IQ labels, have been determined.

4.1.5 Creating the Interaction Quality Corpus

While we have proposed several novel approaches for estimating the Interaction Quality of a dialogue—both static and sequential—, nothing has been said about their performance. To evaluate our approaches, experiments have been conducted using the same data. This is important as producing comparable results is crucial. However, simply collecting dialogue data (including the interaction parameters) is not sufficient as the IQ labels—the ground truth—also needs to be annotated. Here, expert raters may be used following a strict annotation procedure. In the following,

the data used for all experiments on IQ estimation is presented. The data comprises an already existing corpus as well as a newly created extension. The latter has been specially created and annotated by us for our experiments and is publicly available[6]. For the extended corpus, we have added a detailed description of how the IQ labels have been annotated and the final ground truth has been derived from the single expert ratings.

The original corpus associated with the Interaction Quality paradigm is the *LEGO* corpus (Schmitt et al., 2012) based on task-oriented human-machine-dialogues within a single domain. It consists of calls to the Let's Go Bus Information System of the Carnegie Mellon University in Pittsburgh (Raux et al., 2005) which has been online since 2005. People are redirected to Let's Go if they call outside the operating hours of the human-operated service of the bus operating company to get information about the bus schedule in Pittsburgh. The domain consists of four slots, which are the departure place, the destination, the departure time and optionally the bus route. Table 4.1 shows an overview of the different slots and the number of possible values for each slot to give a better impression of the complexity of the domain.

The *LEGO* corpus contains 200 calls to Let's Go from 2006. These calls consist of 4,885 system-user-exchanges annotated with IQ by three different expert raters. They achieved an inter-rater-agreement of $\kappa = 0.54$ using Cohen's Weighted Kappa[7] (Cohen, 1960, 1968). An excerpt of an example dialogue from *LEGO* including annotated final IQ values is depicted in Figure 4.2. In order to extend the *LEGO* corpus, an additional 201 calls to the Let's Go Bus Information System from 2007 consisting of 4,753 exchanges have been annotated to constitute the *LEGOext* corpus (Ultes et al., 2015b). Again, three different raters being advanced students of computer science were asked to annotate each system-user-exchange with one out of five satisfaction labels resulting in an inter-rater agreement of $\kappa = 0.5$.

In the following, we will present in detail how the expert raters have annotated the dialogues to generate the ground truth for the *LEGOext* corpus. The procedure was almost identical to annotating *LEGO*.

Annotation Procedure

For the annotation of the *LEGOext* corpus, the expert raters were required to follow a strict procedure. To acquire IQ ratings, a web-based form was used (Figure 4.12). By that, providing the labelling environment to the raters was very easy. The form was designed to provide all necessary information the raters need to annotated the IQ labels. Hence, the complete dialogue has been displayed with one system-user-exchange per line. While the system utterance could be provided easily, this was different for the user turn: here, the ASR recognition result was used. As ASR is known to not provide correct results, the audio recording for each user turn was also available. Furthermore, as misalignments of user input and system output may occur, e.g., the user starts talking while the system still talks, the audio recording of the complete call was also available. By that, the raters had all information necessary to make a profound decision.

Following the same rating guidelines as in the original *LEGO* corpus (Schmitt et al., 2012), the three raters achieved an overall inter-rater agreement of $\kappa = 0.5$. The guidelines have been established during the annotation process of the *LEGO* corpus to restrict the variations inherit

[6] The may be downloaded from `http://nt.uni-ulm.de/ds-lego`.

[7] The κ is linearly weighted according to the distance between the numerical IQ values. See Section 2.2.3 for more details.

Table 4.1: User dialogue act categories (slots) as defined in the Let's Go system (Thomson et al., 2010) along with their size, i.e., the number of different values per slot.

Slot	Category	Size	Example values
departure place	*from*	3	*ftstop*, *ftmonument*, *ftneigh*
	fstop	328 455	"FORBES&MURRAY", "ANYWHERE&FORBES"
	fmon	52	"AIRPORT", "CENTURY SQUARE"
	fneigh	220	"DOWNTOWN", "SQUIRREL HILL"
destination	*to*	3	*tstop*, *tmonument*, *tneigh*
	tstop	328 455	"FORBES&MURRAY", "ANYWHERE&FORBES"
	tmon	52	"AIRPORT", "CENTURY SQUARE"
	tneigh	220	"DOWNTOWN", "SQUIRREL HILL"
departure time	*time*	2	"NEXT", time specific
	hour	12	"ONE", "TWELVE"
	min	60	"ZERO", "TEN", "THIRTY FIVE"
	pd	2	"AM", "PM"
	day	2	day 10, "TODAY", "WEDNEDAY", "TOMORROW"
	tref	4	"ARRIVE BEFORE", "LEAVE AFTER"
bus route	*bus*	101	"64A", "28X"
meta	*meth*	4	"RESTART", "FINISHED", constraints,
	disc	9	"REPEAT", "FOLLOWING", "PREVIOUS", none

in subjective ratings but allowing enough flexibility for the raters to express their opinion at the same time. These rating guidelines are presented in Figure B.1.

Comparing the agreement κ and correlation ρ of the individual IQ ratings between the two corpora depicted in Table 4.2 shows that the annotation process using the guidelines results in similar agreement.

Table 4.2: Agreement (κ) and correlation (ρ) in IQ ratings of the three raters in *LEGO* and *LEGOext* originally published in (Ultes et al., 2015b). Expert ratings show similar correlations among each other.

LEGOext

	R1/R2	R1/R3	R2/R3	Mean
κ	.59	.51	.40	**.50**
ρ	.73	.66	.67	**.69**

LEGO

	R1/R2	R1/R3	R2/R3	Mean
κ	.64	.48	.51	**.54**
ρ	.79	.68	.70	**.72**

As there have been several annotations (from several raters) per exchange, these are combined to one final label in order to be used for classification. This process is described in the following.

Label Definition—Candidate Selection

Since the aim is to model a general opinion on Interaction Quality, i.e., mirroring the IQ score other raters (and eventually users) agree with, the final label is determined empirically. Majority

Call ID: 2070617000

Turn	Prompt	Audio	Interaction Quality				
			5	4	3	2	1
0	Welcome to the CMU Let's Go bus information system.		◉	○	○	○	○
1	To get help at any time, just say Help or press zero.		◉	○	○	○	○
2	What can I do for you?	▶	◉	○	○	○	○
3	I am an automated spoken dialogue system that can give you schedule information for bus routes in Pittsburgh's East End. You can ask me about the following buses: 28X, 54C, 56U, 59U, 61A, 61B, 61C, 61D, 61F, 64A, 69A, and 501.		◉	○	○	○	○
4	What bus schedule information are you looking for?		◉	○	○	○	○
5	For example, you can say, when is the next 28X from DOWNTOWN to THE AIRPORT? or I'd like to go from MCKEESPORT to HOMESTEAD tomorrow at 10 a.m..	▶	◉	○	○	○	○
6	Leaving from DYNAMO WAY . Is this correct?	▶	◉	○	○	○	○
7	Leaving from DYNAMO WAY . Is this correct?	▶	◉	○	○	○	○
8	Okay		◉	○	○	○	○
9	Going to SQUIRREL HILL . Is this correct?	▶	◉	○	○	○	○
10	Right		◉	○	○	○	○
11	At what time do you want to travel?	▶	◉	○	○	○	○
12	Leaving at 6 a.m. . Did I get that right?	▶	◉	○	○	○	○
13	Okay		◉	○	○	○	○
14	Hold on. I'll look that up.		◉	○	○	○	○
15	There is a 61A leaving DYNAMO WAY AT BEECH at 6 oh 2 a.m. . It will arrive at FORBES AVENUE AT MURRAY at 6 38 a.m. .		◉	○	○	○	○
16	To get more information about buses related to this trip, you can say, when is the next bus, or, when is the previous bus. To ask about a different trip, you can say, start a new query. If you are finished, you can say goodbye.	▶	◉	○	○	○	○

Fig. 4.12: The online labelling form used by expert raters for annotating the *LEGOext* corpus originally published in (Ultes et al., 2015b).

voting for deriving the final IQ label is not applicable since many exchanges are labelled with three different ratings, i.e., each of the three raters opted for a different score, thus forming no majority for either score. Therefore, the mean of all rater opinions is considered as possible candidate for the final class label:

$$rating_{mean} = \lfloor \left(\frac{1}{R} \sum_{r=1}^{R} IQ_r \right) + 0.5 \rfloor \, . \tag{4.19}$$

Here, IQ_r is the Interaction Quality score provided by rater r. $\lfloor y \rfloor$ denotes the highest integer value smaller than y. Every value IQ_r contributes equally to the result that is finally rounded to the closest integer value.

Furthermore, the median is considered, which is defined as

$$rating_{median} = select(sort(IQ_r), \frac{R+1}{2}) \, , \tag{4.20}$$

where *sort* is a function that orders the ratings IQ_r of all R raters ascendingly and $select(list, i)$ chooses the item with index i from the list *list*. In other words, the IQ score separating the higher half of all ratings to the lower half is selected as final IQ score.

Table 4.3 shows the agreement between the mean and median labels with the single user ratings. Clearly, the median represents the better choice for deriving the final label given the

higher values in κ, ρ, and UAR. This validates the findings for the original experiments in the *LEGO* corpus.

Table 4.3: Agreement of single rater opinions to the merged label when determined by mean and median, measured in UAR, κ, and ρ originally published in (Ultes et al., 2015b). On the left side is *LEGOext*, on the right side *LEGO*.

LEGOext

	Mean Label	**Median Label**
UAR		
Rater1	.657	.839
Rater2	.412	.660
Rater3	.556	.450
Mean	.542	**.650**
Cohen's weighted κ		
Rater1	.765	.878
Rater2	.697	.691
Rater3	.630	.655
Mean	.697	**.741**
Spearman's ρ		
Rater1	.843	.891
Rater2	.905	.846
Rater3	.782	.799
Mean	.843	**.845**

LEGO

	Mean Label	**Median Label**
UAR		
Rater1	.653	.751
Rater2	.647	.735
Rater3	.516	.550
Mean	.605	**.679**
Cohen's weighted κ		
Rater1	.763	.815
Rater2	.767	.814
Rater3	.657	.658
Mean	.729	**.762**
Spearman's ρ		
Rater1	.901	.900
Rater2	.911	.907
Rater3	.841	.814
Mean	**.884**	.874

Hence, the chosen methods are used to create the final corpora *LEGO* and *LEGOext*. Details on their statistics are shown in the next section.

Corpus Statistics

The distribution of the final IQ label is shown in Figure 4.13. For the *LEGOext* corpus, label "satisfied" (5) has been assigned much more frequently while all others have been assigned less often compared to the *LEGO* corpus. This increase in overall system performance may be a result of an improved system as the 2007 version of Let's Go represents an updated system.

Naturally, this performance increase also results in a higher average IQ score for the *LEGOext* corpus: it achieves an average IQ of 4.46 while the *LEGO* corpus achieves 3.39 averaged over all labelled system-user exchanges.

The statistics for both corpora as well as for the combined data set *LEGOv2* are depicted in the following table originally published in (Ultes et al., 2015b):

Corpus	**Year**	**#calls**	**#exchanges**	**avg. Length**	κ	**avg. IQ**
LEGO	2006	200	4,885	25.4	.54	3.39
LEGOext	2007	201	4,753	22.6	.50	4.46
LEGOv2		401	9,638	24.0	.52	4.46

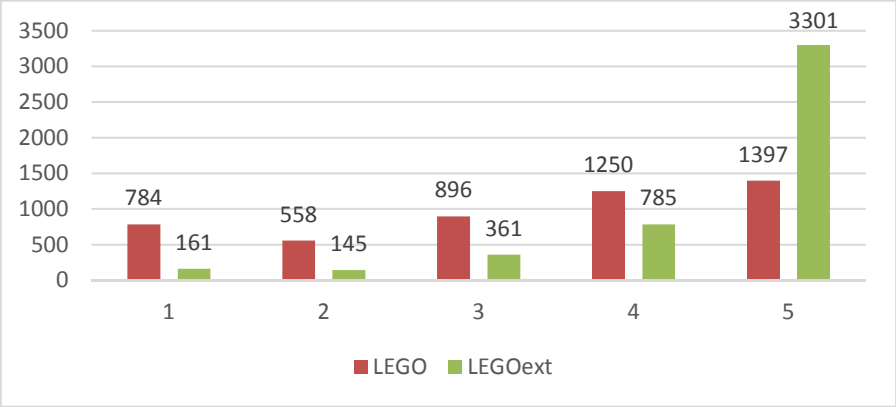


Fig. 4.13: The distribution of the final label scores along with there absolute number of occurrences for the *LEGO* and the *LEGOext* corpus originally published in (Ultes et al., 2015b)

Having not only the implementation but also corpus data consisting of example dialogues with extracted interaction parameters and annotated Interaction Quality values finally allows for comparable experiments evaluating the estimation approaches presented in Sections 4.1.2 and 4.1.3. The evaluation procedure and the results will be presented in the following section.

4.1.6 Evaluation of Approaches for Interaction Quality Recognition

While we have presented our own ideas and approaches for estimating the Interaction Quality and have also presented our newly created corpus used for evaluation data and how we have defined the ground truth, in this section, we will present the setup and the results of the evaluation of those approaches. First, we will give a description of the different sets of features used as input to the statistical classifiers. As some approaches have different requirements on the characteristics of the features, different sets have been created. Those are then applied to the proposed estimation approaches. The results of these experiments are presented including a thorough discussion.

Feature Set Definition

The evaluation of the presented IQ estimation approaches is based on the *LEGO* and *LEGOext* corpora (Schmitt et al., 2012; Ultes et al., 2015b) which define a set of interaction parameters. These 51 parameters are automatically derivable textual and numerical values and contain domain-specific as well as domain-independent features. As the presented approaches have different requirements on the characteristics of the features used as input variables, we define three feature sets. The BASE*all* feature set contains all 51 parameters listed in Table B.1.

However, some parameters contain text values whereas some classification approaches require the input variables to be strictly numerical. As the textual parameters have a high number of possible values, encoding those as numericals would not make sense. Furthermore, these parameters are deemed to be very domain domain-specific as well. Hence, the domain-independent feature set BASE*di* is defined containing 46 numerical parameters.

Moreover, for applying the JaHMM and JaCHMM libraries, covariance matrices are created which must be invertible. As some features do not change over one call, e.g., if the user does not

request help, the covariance matrix would contain a row of zeroes and thus lose the property of being invertible. To avoid this, a third feature set is created—BASE*25*—containing 25 features. Those have been selected empirically.

A summary of all feature sets is depicted in Table 4.4 and the parameters for each feature set is listed in Table B.2. Consequently, the defined feature sets are applied for all approaches for IQ estimation which requirements are met. The results will be presented and discussed in the following.

Table 4.4: The definition of feature sets used for experiments on automatic recognition of Interaction Quality.

ID	# features	value range	domain-independence
BASE*25*	25	numerical	yes
BASE*di*	46	numerical	yes
BASE*all*	51	textual & numerical	no

The Baseline: Standard Machine Learning Methods

Previous work on IQ estimation relied on SVMs based on SMO for creating the statistical model (Schmitt et al., 2011a; Schmitt and Ultes, 2015). As they used only a subset of the available data for evaluation, we repeated the SVM experiments to create a baseline for our experiments. Moreover, we are not only applying SVM estimation. As we are further interested how classifiers of different characteristics perform, all standard machine learning approaches—which more elaborated methods are based on—form our baselines (see Sec. 4.1.2). As they are able to handle all types of features, all feature sets are used for evaluation. The results are displayed in Table 4.5. These experiments have been conducted using data only from the *LEGO* corpus in a 10-fold cross-validation setting to ensure consistent results. For evaluation, the four metrics Unweighted Average Recall (UAR), Cohen's weighted κ, Spearman's ρ, and the root mean squared error (RMSE) are shown[8].

Clearly, the Rule Learner (RL) achieves the overall best performance in all four metrics for BASE*di* and BASE*25*. Only for ρ of BASE*all* it is beaten by the SVM. The overall performance of the Naïve Bayes classifier was the worst for all feature sets. Evidently, the problem of IQ estimation may not be modelled using such a simple classification approach. As Naïve Bayes does not show good performance, it will not be regarded any further within this part of IQ estimation evaluation.

It is interesting to see that the original baseline—the SVM with SMO—only performs second best for almost all feature sets and metrics and that instead a rule learner provides better results. The question is if this means that the problem of IQ estimation is rather simple and easy rules can be applied. To answer this question, we have analysed the rules in more detail (Ultes and Minker, 2013c): one rule handles an average of 17 samples. Moreover, only 22 rules per fold cover more than or equal to 30 exchanges while 289 cover less than 30 exchanges. This shows that there are

[8] For a detailed explanation of these metrics, see Section 2.2.3.

Table 4.5: Results for the standard static ML approaches for all three feature sets. The rule learner achieves best performance having the SVM at the second place.

	UAR	κ	ρ	RMSE
	BASEall			
SVM (SMO)	0.501	0.614	0.764	0.974
Naïve Bayes	0.409	0.481	0.654	1.269
Rule Learner	0.560	0.642	0.756	0.934
	BASEdi			
SVM (SMO)	0.483	0.604	0.758	0.994
SVM (libSVM)	0.471	0.582	0.725	1.018
Naïve Bayes	0.396	0.461	0.638	1.283
Rule Learner	0.602	0.685	0.794	0.883
	BASE25			
SVM (SMO)	0.469	0.590	0.752	1.015
Naïve Bayes	0.422	0.516	0.685	1.126
Rule Learner	0.588	0.671	0.781	0.905

a high number of rules which cover outliers leading to the assumption that a rule learner does not generalise well across data sets.

To test this hypothesis, cross-corpus experiments have been conducted where a rule learner has been trained on *LEGO* and evaluated on *LEGOext* and vice versa (Ultes et al., 2015b). For comparison reasons, the same experiments have also been conducted using a SVM (SMO). The results in Table 4.6 show that the SVM performs better for both cases validating that the rule learner's ability to generalise are limited.

Table 4.6: Cross-corpus experiments for RL and SVM trained on one corpus and evaluated on the other. The SVM clearly provides better cross-corpus performance.

	Train	Eval	UAR	κ	ρ	RMSE
Rule Learner	*LEGO*	*LEGOext*	0.327	0.265	0.437	1.315
	LEGOext	*LEGO*	0.288	0.235	0.513	1.641
SVM (SMO)	*LEGO*	*LEGOext*	0.409	0.379	0.554	1.200
	LEGOext	*LEGO*	0.326	0.322	0.558	1.552

For a deeper analysis of the cross-corpus performance of the SVM (SMO), each feature group, i.e., the group of features which originate from the same dialogue system module, has been investigated separately. The results are depicted in Figure 4.14. While the general performance is lower compared to in-corpus classification, the results are clearly above the majority baseline[9] for

[9] Majority baseline means that the majority class is always predicted. This would result in an UAR of 0.2 for a five class problem.

all feature groups. While the best performance is achieved using all features, the DM parameters contribute most to the generalisation ability of the IQ paradigm.

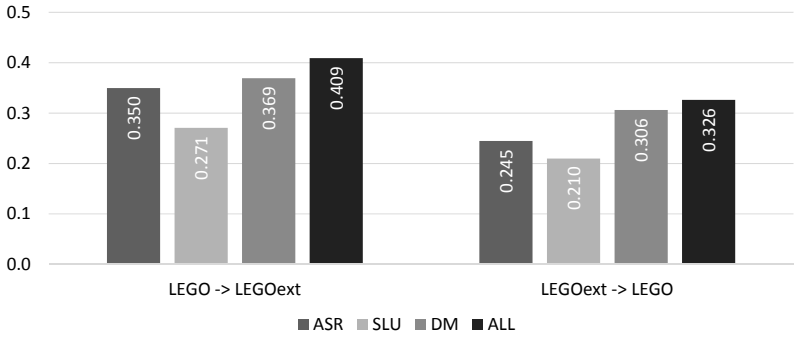


Fig. 4.14: Results of cross-corpus SVM classification for each feature group (ASR, SLU, DM, ALL) separately.

Comparing the performance measures on the different feature sets for all classifiers shows that for the SVM, the more features are used the better results are achieved. However, the differences are only minor meaning that the feature reduction represented by BASE*di* and BASE*25* is quite reasonable and not a lot of information is lost (at least for the given data set). This is also shown by the performances of the rule learner which even achieves worst performance using all features. This may be a result of non-numerical features present in BASE*all*.

Finally, for BASE*di*, two implementations of the SVM have been investigated. This is due to the fact that the implementation of the SMO does not provide usable confidence values[10] which we need for the confidence-based error correction and the Hybrid-HMM. Here, the libSVM produces better values. Unfortunately, this switch is accompanied by a slight loss in performance which may be attributed to the additional optimisation within the SMO algorithm.

The performance of the presented static classification approaches for the different feature sets are used as the baseline for all following experiments. Furthermore, due to the plug-in nature of the Hybrid-HMM and the error correction, the results are there of special importance.

Hierarchical Error Correction

Based on the standard machine learning approaches, the hierarchical error correction approach (Sec. 4.1.2) has been evaluated. For stage one, the results computed on the *LEGO* corpus using BASE*di* as presented in the previous section are used. Based on those results, the error has been calculated and used to train the error model. The performances of the two statistical classification methods Rule Learner (RL) and SVM are compared, both applied for stage one and stage two. However, for stage two, the libSVM implementation has been used. This is as the libSVM is able to create a reasonable model, i.e., not always assigning the majority class, even if one class dominates. Furthermore, a normalisation component is added performing a range normalisation of the input parameters. This is necessary for using the implementation of the statistical classification algorithms at hand.

[10] The SMO computes for a five class problem always the confidence measures 0.4, 0.3, 0.2, 0.1, and 0.0.

For error estimation, two variants are explored: using one combined model for all three error classes (−1,0,+1) and using two separate models, one for distinguishing between −1 and 0 and one for distinguishing between +1 and 0 with combining their results afterwards. While using RL for error estimation yields reasonable performance results for the combined model, it is not suitable for error estimation using two separate models as all input vectors are mapped to 0. Hence, for the two model approach, only the SVM is applied.

Table 4.7: The results for error correction (using the bounded difference) and the simple hierarchical approach. The first column shows the classification algorithm used for stage one while the second column shows the algorithm of the second stage.

		UAR	κ	ρ	RMSE
	Error Correction (h^{diff})				
SVM (SMO)	SVM (libSVM)	0.472	0.597	0.754	0.983
	Rule Learner	0.524	0.646	0.788	0.904
	2xSVM (linear)	0.492	0.621	0.778	0.902
Rule Learner	SVM (libSVM)	0.573	0.687	0.811	0.828
	Rule Learner	0.562	0.674	0.801	0.849
	2xSVM (linear)	0.606	0.687	0.800	0.855
	Simple Hierarchical Approach				
SVM (SMO)	SVM (libSVM)	0.482	0.604	0.758	0.996
	Rule Learner	0.585	0.674	0.792	0.891
Rule Learner	SVM (libSVM)	0.565	0.688	0.816	0.852
	Rule Learner	0.584	0.680	0.798	0.871

Results for applying error correction (EC) calculating h_f^{diff} (see Eq. 4.6) are presented in Table 4.7 and compared to the simple hierarchical approach. First, we analyse the error correction results compared to the one-stage results of Section 4.1.6. The relative improvements are depicted in Figure 4.15. As can be seen, error correction may improve performance having either an SVM or RL at stage one. Effects on the SVM performance are much stronger than on the rule learner for all four metrics. While the effects of 2xSVM are not as strong as applying RL at stage two when having the SVM at stage one, it may be considered as the overall best performing approach as it achieves best results also improving RL performance. All differences are statistically significant ($p < 0.001$), tested with the Wilcoxon test (Wilcoxon, 1945).

Furthermore, these results are compared to a simple hierarchical approach (SH) where the hypothesis h_0 of the stage one classifier is used as an additional feature for the stage two classifier targeting IQ directly (see Sec. 4.1.2). Here, the performance of the stage two classifier is of most interest since this approach can be viewed as one stage classification with an additional feature. The results in Table 4.7 show that RL does not benefit from additional information (comparison of respective last rows with one stage RL recognition of Table 4.5). SVM recognition at stage two, though, shows better results. While its performance is reduced using the SVM hypothesis as additional feature, adding the RL hypothesis improved UAR up to 17% relatively. However, there

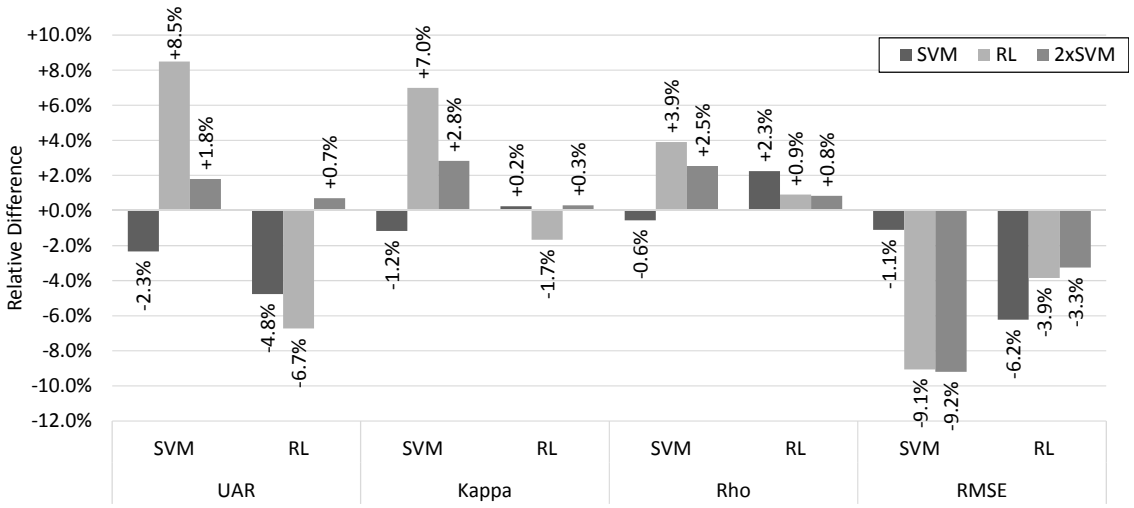


Fig. 4.15: Relative difference of Error Correction results using the bounded difference. The relative results are computed against the results of the BASE*di* feature set in Section 4.1.6. The horizontal axis represents the classifier at stage one grouped by the evaluation metrics.

is no reasonable scenario where one would not use the better performing RL in favour of using its results as additional input for SVM recognition.

The question remains why SVM benefits from Error Correction as well as from adding additional input parameters while RL does not. It remains unclear if this is an effect of the task characteristics combined with the characteristics of the classification method. It may as well be caused by low classification performance. A classifier with low performance might be more likely to improve its performance by additional information or EC.

As the error correction approach using 2xSVM at stage two has been identified as the overall best performing approach, we will only consider this approach for further analysis of confidence-based error correction. Moreover, as SVMs have been shown to generalise better, only SVMs for stage one are used. Here, the libSVM implementation has been used as it is crucial for the confidence model to have usable confidence values of the stage one classifier. Again, the results were obtained in a cross-validation setting.

Table 4.8 shows the results of using the confidences models for deriving the final IQ hypothesis based on Equations 4.14 and 4.15. While using the bounded difference to compute the final hypothesis h_f^{diff} achieves much better results for the SVM based on libSVM, h_f^{max} further improves the estimation performance. Figure 4.16 shows the relative differences between h_f^{max} and h_f^{diff}, and h_f^{max} and h_0^{max} (the performance of the libSVM at stage one). Using the IQ value with the maximum confidence for the final hypothesis clearly outperforms the bounded difference in all four metrics. These differences are even significant ($p < 0.001$ with Wilcoxon test (Wilcoxon, 1945)).

To derive the final hypothesis, not only the maximum confidence may be used but also the weighted sum over all confidences as defined by Equation 4.15. Here, the same results as for h_f^{diff} are used and only the confidences have been calculated additionally. The relative performance

Table 4.8: Results of error correction having a linear SVM at stage one and two linear SVMs at stage two. Here, different methods for deriving the final hypothesis are compared.

Stage 1	Stage 2		UAR	κ	ρ	RMSE
SVM (libSVM)	*none*	h_0^{max}	0.471	0.582	0.725	1.036
		h_0^{wSum}	0.468	0.584	0.749	0.890
	2xSVM (libSVM)	h_f^{diff}	0.507	0.633	0.782	0.862
		h_f^{max}	0.517	0.646	0.798	0.830
		h_f^{wSum}	0.409	0.511	0.775	0.899

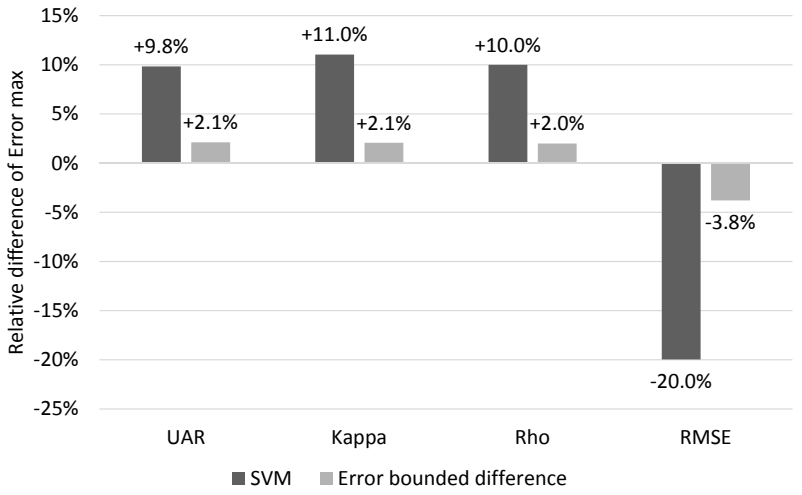


Fig. 4.16: Relative difference of h_f^{max} to h_0^{max} (SVM) and h_f^{diff} (error bounded difference).

gain by using h_f^{wSum} compared to the bounded difference h_f^{diff}, the maximum confidence h_f^{max}, and computing the weighted sum for the initial SVM results h_0^{wSum} is presented in Figure 4.17.

The comparisons show that using a weighted sum to generate the final hypothesis results in a decrease for almost all metrics: UAR drops by at least 12.6% and κ by at least 12.5%. ρ and RMSE show similar effects although not with the same magnitued. Hence, using a weighted sum does not result in an increase in recognition performance.

While only evaluation of static approaches for IQ estimation have been described so far, we are also interested in how the sequential approaches perform. Here, an emphasis is laid on the Hybrid-HMM approach having the static approaches behave in a sequential setting. All of this will be thoroughly evaluated and analysed in the following.

Hidden Markov Model-based Approaches

For evaluating the HMM and CHMM approaches described in Section 4.1.3, the feature set BASE*25* is used. Due to the special requirements of the models, domain-related, textual or quasi-constant features are removed as this would have resulted in rows of zeros during computation of the covariance matrices of the feature vectors. A row of zeros in the covariance matrix will make it non-invertible, which will cause errors during the computation of the emission probabilities.

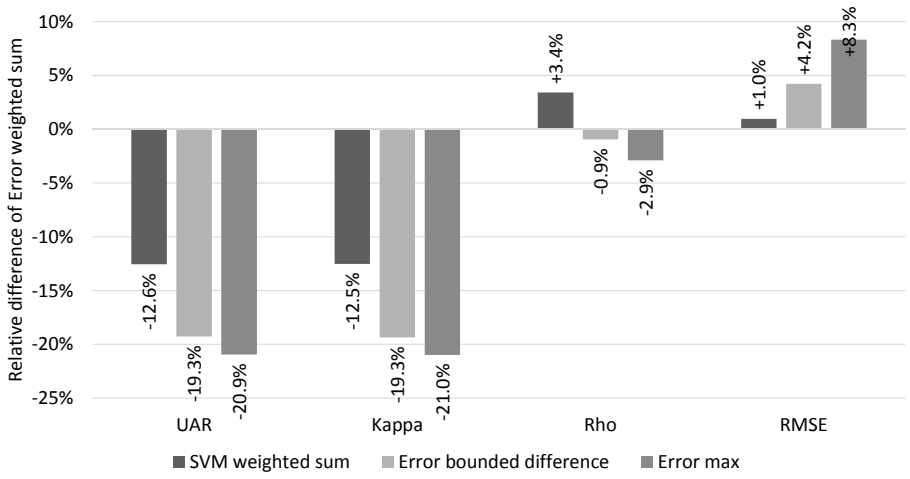


Fig. 4.17: Relative difference of h_f^{wSum} to h_0^{wSum} (SVM weighted sum), h_f^{diff} (error bounded difference), and h_f^{max} (error max).

Results of the experiments are presented in Table 4.9 ranked according to the number of hidden states used for the CHMM. The accuracy of the CHMM decreased remarkably after passing the threshold of 9 states, where the highest values for UAR, κ, and ρ could be achieved (Ultes et al., 2012a).

The results are computed using 6-fold cross validation on the dialogue-level. Best performance for the CHMM was achieved for nine states with an UAR of 0.39, Cohen's κ of 0.43, and Spearman's ρ of 0.60. Applying the HMM using five hidden states, 6-fold cross validation resulted in an UAR of 0.44, κ of 0.56, and ρ of 0.72 this clearly outperforming the best CHMM configuration.

Comparing the results of the HMM and the CHMM with the SVM baseline (see Sec. 4.1.6) presented in Figure 4.18 clearly shows that both Markov model approaches were not able to outperform the baseline. One possible reason for this is may be a lack of training data. For the CHMM, for 9 hidden states, a total of 8,280 parameters (initial probabilities, transition probabilities, and mean and covariance matrices of the emission probabilities) have to be learned. Calculating these with a total of 3,908 training vectors (per fold) results in less than one vector per parameter on average. This has shown to be insufficient training data to create proper estimates for the parameters.

In order to further evaluate the CHMM on this task, additional experiments have been conducted with the complete data of the *LEGOv2* corpus. However, the results are similar to the ones obtained on the *LEGO* corpus (Ultes et al., 2015b). Hence, it is unlikely that the amount of data is the reason for the performance drop. Another possible reason for the performance of the HMM and the CHMM is the way the observation probability is modelled. For both Markov models, the observation probability is defined by Gaussian mixture models. However, replacing GMMs with confidence scores may result in better performance.

Hence, in contrast to the HMM where the observation probability is modelled using GMMs, the Hybrid HMM uses confidence scores of static classifiers instead (see Sec. 4.1.3). More precisely, an SVM (SMO and libSVM implementations) and a Rule Learner have been used to

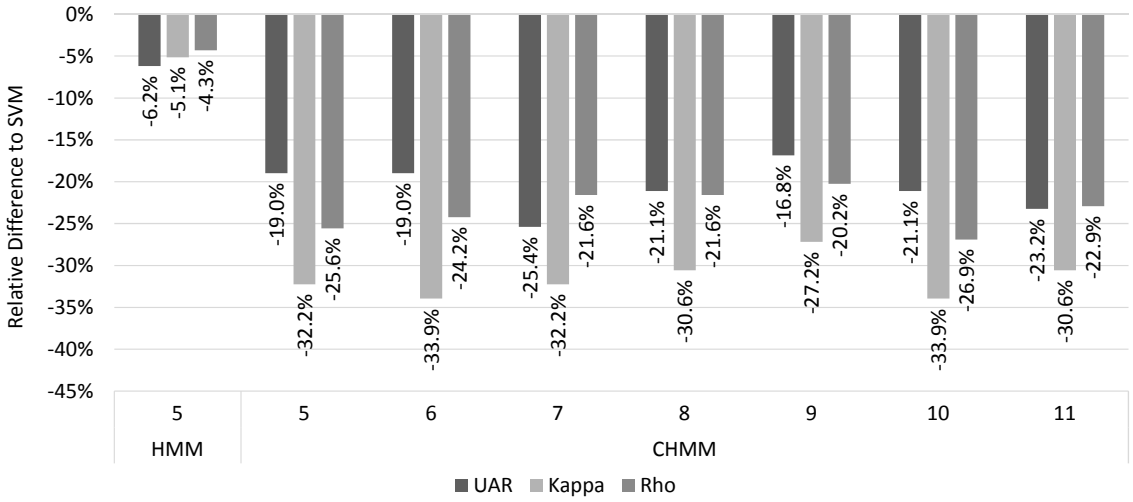


Fig. 4.18: The relative performance change of HMM and CHMM compared to the SVM baseline. Clearly, applying both approaches results in a loss in performance.

Table 4.9: Results for HMM experiment according to the number of hidden states along with results for regular HMM and SVM classification originally published in (Ultes et al., 2012a). The '$*$' indicates the best result.

	# states	**UAR**	**Kappa**	**Rho**
HMM	5	0.44	0.56	0.72
	5	0.38	0.40	0.56
	6	0.38	0.39	0.57
	7	0.35	0.40	0.59
CHMM	8	0.37	0.41	0.59
	9*	0.39	0.43	0.60
	10	0.37	0.39	0.55
	11	0.36	0.41	0.58

produce the confidence scores. Additionally, the introduction of a confidence model into the hierarchical error correction approach makes error correction also usable for the Hybrid HMM. Here, only having an SVM at stage one and two SVMs at stage two (discriminating between the error classes +1 vs. 0 and -1 vs. 0) is regarded as this configuration showed to achieve the overall best performance.

Results for applying all those approaches in a cross-validation setting using the BASE*di* feature set are depicted in Table 4.10. Here, the conventional method of deriving the final hypothesis as described in Equation 4.17 (which is equivalent to Eq. 4.14) has been applied. For computing the transition probability, an action-independent (AI) and an action-dependent (AD) transition model has been applied as described in Section 4.1.3 by calculating the transition frequencies in the training data. Here, not the actual actions have been used but the type of action. This information is represented by the interaction parameter ACTIVITYTYPE (see Table B.1). The parameter

Table 4.10: Results of the Hybrid HMM using the maximum probability for deriving the final hypothesis for different ways of modelling the transition probability: action-independent (AI), action-dependent (AD) or with a handcrafted action-independent transition matrix (HC1 and HC2).

	UAR	κ	ρ	RMSE
		AI		
SVM (SMO)	0.486	0.595	0.767	1.060
SVM (libSVM)	0.484	0.586	0.735	1.032
Rule Learner	0.597	0.675	0.800	0.929
Error Correction (2xSVM)	0.494	0.567	0.785	0.991
		AD		
SVM (SMO)	0.473	0.608	0.774	0.923
SVM (libSVM)	0.452	0.559	0.713	1.079
Rule Learner	0.623	0.708	0.824	0.827
Error Correction (2xSVM)	0.505	0.630	0.788	0.852
		HC		
SVM (SMO)	0.502	0.633	0.785	0.908
SVM (libSVM)	0.461	0.570	0.720	1.075
Rule Learner	0.625	0.709	0.824	0.828
Error Correction (2xSVM)	0.521	0.641	0.801	0.856
		HC2		
SVM (SMO)	0.440	0.541	0.752	0.927
SVM (libSVM)	0.478	0.580	0.725	1.013
Rule Learner	0.624	0.706	0.823	0.824
Error Correction (2xSVM)	0.499	0.606	0.793	0.856

may take one out of four values: 'Question', 'Announcement', 'Confirmation', and 'wait'. Hence, a new transition probability matrix has been computed for each of the four action types.

Additionally to the above mentioned configurations, two handcrafted transition matrices have been applied for the AI variant which are presented in Table 4.11. Both are designed to only allow transitions between adjacent IQ labels. The first handcrafted matrix (HC1) has a higher probability of staying in IQ value 5 or 1 than transitioning to another IQ value compared to the IQ values 2, 3, and 4 while the second matrix (HC2) has always the same probability of remaining in the same IQ value or transitioning to another.

The results for action-dependent and action-independent transition probabilities are compared in Figure 4.19. Unfortunately, the overall performance could not be improved: neither for AI nor for AD, an improvement in *all* metrics for *all* classifiers could be found[11] compared to applying only the base classifier. By looking at the individual classifiers more closely, though, shows that for the SVM (libSVM), the performance could be improved for AI in all metrics. Furthermore, the performance using the Rule Learner could also be improved for AD in all metrics.

[11] For UAR, κ, and ρ, an increase means improvement while for RMSE, a decrease means improvement.

Table 4.11: The two handcrafted transition matrices HC1 and HC2 for modelling the transition probability P_t based on empirical data. Both only differ in the transitions from 1 to 1 and 2 and from 5 to 4 and 5 (HC1 / HC2).

from \ to	1	2	3	4	5
1	0.7/0.5	0.3/0.5	0	0	0
2	0.25	0.5	0.25	0	0
3	0	0.25	0.5	0.25	0
4	0	0	0.25	0.5	0.25
5	0	0	0	0.3/0.5	0.7/0.5

When comparing AD with AI, it becomes clear that for the base classifiers RL and EC, AD achieves better performance measures in all metrics. This overall pattern of having better performance when adding action dependency is not true for the SVM classifiers. Those are the only approaches where AI performed better than AD. The difference between AI and AD is significant for all classifiers and has been tested using the Wilcoxon test (Wilcoxon, 1945). As already described, not the actual system actions have been used but the type of system action, i.e., the interaction parameter ACTIVITYTYPE.

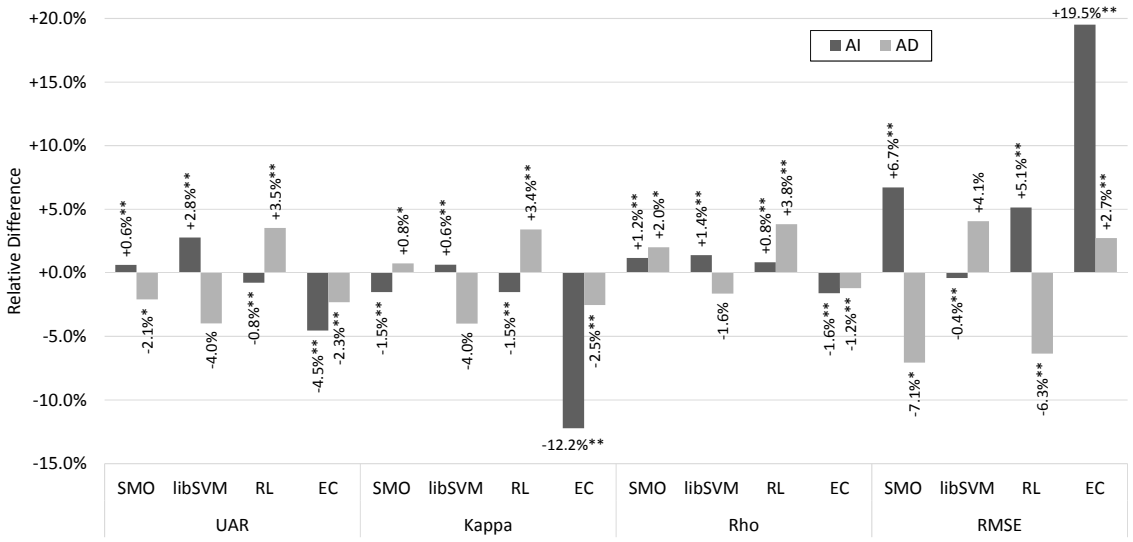


Fig. 4.19: The relative difference in performance to the base classifier of the Hybrid-HMM using action-dependent (AD) and action-independent (AI) transition probabilities. Significance (**: $p < 0.001$, *: $p < 0.05$) has been tested with the Wilcoxon test (Wilcoxon, 1945).

Comparing the absolute performances of HC1 and HC2 with AI and AD shows that using a handcrafted transition matrix may result in an overall better performance than frequency-based transition matrices. However, which of the two proposed handcrafted matrices (HC1 or HC2) results in better performance is not clear and depends on the base classifier used. Hence, comparing

the relative differences shown in Figure 4.20, HC2 shows higher improvement for the libSVM but HC1 shows better performance for all other classifiers.

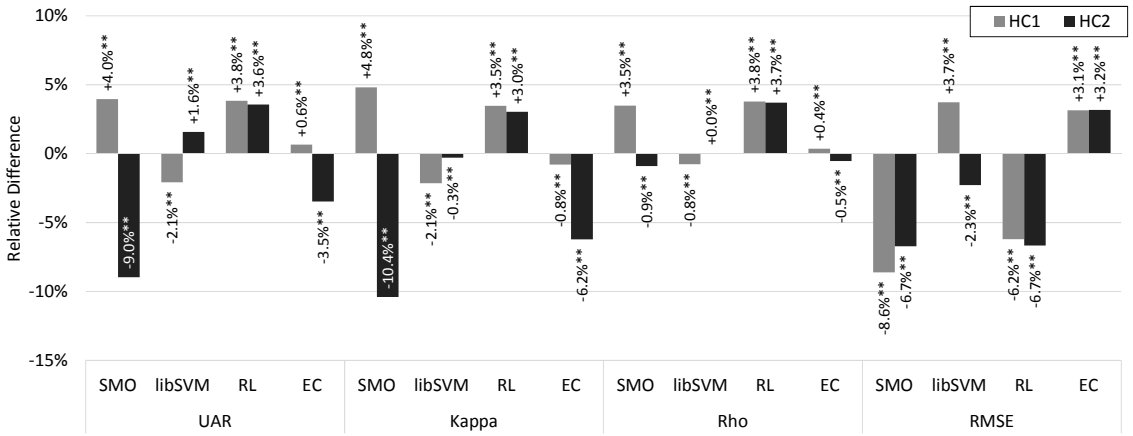


Fig. 4.20: The relative difference in performance to the base classifier of the Hybrid-HMM using handcrafted action-independent transition probabilities with different transition matrices (HC1 and HC2). Significance (**: $p < 0.001$) has been tested with the Wilcoxon test (Wilcoxon, 1945).

The overall best performance for applying a Hybrid-HMM could be achieved using a rule learner as observation probability provider achieving an UAR of 0.625 (HC1), a κ of 0.709 (HC1), a ρ of 0.824 (AD,HC1) and an RMSE of 0.824 (HC2). All these results also present an improvement compared to plain RL performance for BASE*di*. However, as already discussed in Section 4.1.6, applying a rule learner does not generalise as well as applying an SVM-based approach. If we look at the remaining SVM-based approaches, applying Error Correction for providing the observation probability model seems to be the best performing approach with an UAR of 0.521 (HC1), a κ of 0.641 (HC1), a ρ of 0.801 (HC1), and an RMSE of 0.852 (AD). Again, all these values represent a significant performance improvement compared to only applying error correction ($p < .0001$, Wilcoxon test (Wilcoxon, 1945)).

Analysing the improvements alone shows that the SMO could benefit most from thy Hybrid-HMM approach among all SVM-based approaches: UAR +4.0% (HC1), κ +4.8% (HC1), ρ +3.5% (HC1), RMSE -8.6% (HC1). This may most likely be attributed to the way how the SMO models its confidence scores. For a five-class problem, the classes get assigned the confidence scores statically according to their order (here from most to least probable): 0.4, 0.3, 0.2, 0.1, 0.0.

While we have only regarded the results of deriving the final hypothesis by taking the IQ value with the maximum probability so far, results for using the weighted sum are shown in Table 4.12. The same phenomena as described previously are also valid for using the weighted sum: the rule learner achieves best overall performance having the error correction approach at second place. What can be noted is that, for some metrics and configurations, the performance is improved (e.g., UAR for Rule Learner (AD, HC1, HC2)), while for others, the performance drops drastically (e.g., UAR for SMO (AI, HC1, HC2)) by using the weighted sum. Having this unclear situation, no profound conclusion may be drawn. The only statement is that the weighted

Table 4.12: Results of the Hybrid HMM using the weighted sum over all state probabilities for deriving the final hypothesis for different ways of modelling the transition probability: action-independent (AI), action-dependent (AD) or with a handcrafted action-independent transition matrix (HC1 and HC2).

	UAR	κ	ρ	RMSE
		AI		
SVM (SMO)	0.409	0.503	0.756	1.021
SVM (libSVM)	0.469	0.550	0.742	0.981
Rule Learner	0.601	0.683	0.815	0.886
Error Correction (2xSVM)	0.442	0.484	0.790	1.027
		AD		
SVM (SMO)	0.485	0.619	0.781	0.888
SVM (libSVM)	0.458	0.572	0.723	1.039
Rule Learner	0.626	0.713	0.831	0.804
Error Correction (2xSVM)	0.504	0.632	0.794	0.840
		HC		
SVM (SMO)	0.461	0.574	0.768	0.897
SVM (libSVM)	0.475	0.586	0.731	1.024
Rule Learner	0.627	0.712	0.830	0.807
Error Correction (2xSVM)	0.520	0.635	0.804	0.843
		HC2		
SVM (SMO)	0.447	0.546	0.765	0.903
SVM (libSVM)	0.483	0.586	0.731	0.979
Rule Learner	0.627	0.710	0.829	0.805
Error Correction (2xSVM)	0.489	0.586	0.792	0.869

sum may be reasonable for some concrete configurations also regarding the goal of the specific application.

For estimating the Interaction Quality of SDS, we have proposed and evaluated several static and sequential novel classification approaches. Before drawing conclusions on IQ estimation, we will continue with a more detailed analysis of the features itself as they already contain temporal information. Bear in mind that this type of information is regarded to be of special importance for a sequential problem like recognising the Interaction Quality

4.1.7 Analysis of Temporal Features

Modelling the Interaction Quality of spoken dialogue interaction may clearly be regarded as a sequential task. However, while several static and sequential approaches have been tested and evaluated, static approaches where very hard to beat by sequential approaches. While static approaches obviously do not reflect the dependence on previous IQ values, they comprise some means of temporal modelling: the set of interaction parameters being modelled on three different levels includes also temporal information. Our hypothesis is that those temporal features play a

major role in the performance lead of static classifiers. This circumstance will be investigated in more detail in the following by an analysis of the temporal interaction parameters.

				%ASR-Success		{#}ASR-Success	
	System DA	User DA	ASR-Status	User	System	User	System
1	[Welcome]	-	-	0.0	0.00	0	0
2	[Help_info]	-	-	0.0	0.00	0	0
3	[Open]	[Inform_origin]	complete	1.0	0.33	1	1
4	[Confirm_origin]	[Inform_origin]	complete	1.0	0.50	2	2
5	[Confirm_origin]	[Affirm]	incomplete	0.66	0.40	2	2
6	[Filler]	-	-	0.66	0.33	2	1
7	[Ask_destination]	[Inform_destination]	complete	0.75	0.43	2	1

Fig. 4.21: The computation of dialogue level parameters %ASR-Success and window-level parameters {#}ASR-Success from the view of the user and the system.

The temporal effects of the dialogue are captured within the interaction parameters by the dialogue and window levels (cf. Sec. 4.1.1). To get a better understanding, the dialogues are analysed manually. The observation been made is that some system-user-exchanges contain only a system turn, e.g., the "Welcome" message of the system (Figure 4.21, line 1). While the system and the user have a different view on the interaction in general, this is especially the case regarding the number of dialogue turns. However, as this information is used for computing parameters on the dialogue and window level, both views should be reflected by the interaction parameters. Hence, the parameters are computed with respect to the number of system turns as well as the number of user turns. The original feature set is extended to contain both variants of parameters.

An example dialogue snippet showing parameters originating from the *ASR-Status* is illustrated in Figure 4.21. It shows both calculation variants for the window parameter *{#}ASR-Success* and the dialogue parameter *%ASR-Success*. The differences are clearly visible: while *%ASR-Success* is either 0 or 1 for the user's view up until line 4 (only successful ASR events occur), the numbers are different for the system's view.

To reflect this system and user view for the complete corpus, a number of parameters are calculated for both variants[12]. The window size remained the same with $n = 3$. This results in an extended feature set consisting of 65 features. For the remainder of the paper, we will refer to the original feature set as BASE*all* and to the extended feature set as EXT*all*.

To get a better understanding of the different parameter level and their contribution to the overall estimation performance, experiments have been conducted using each combination of parameter levels as a feature set, e.g., using only parameters on one level or using parameters from all but one levels. Furthermore, to get a better understanding of the extension of the feature set, the experiments are performed for the original and for the extended corpus. Some interaction parameters with constant value and textual interaction parameters with a task-dependent nature have been discarded resulting in the already known BASE*di* feature set and the new EXT*di* feature

[12] Recalculated parameters: %ASRSuccess, %TimeOutPrompts, %ASRRejections, %TimeOuts_ASRRej, %Barge-Ins, MeanASRConfidence, {#}ASRSuccess, {#}TimeOutPrompts, {#}ASRRejections, {#}TimeOuts_ASRRej, {#}Barge-Ins, {Mean}ASRConfidence

Table 4.13: Results in UAR, κ and ρ for including and excluding different parameter levels for $LEGO_{orig}$ and for $LEGO_{ext}$.

	UAR		κ		ρ	
	$LEGO_{orig}$	$LEGO_{ext}$	$LEGO_{orig}$	$LEGO_{ext}$	$LEGO_{orig}$	$LEGO_{ext}$
only exchange	0.328	0.328	0.310	0.310	0.456	0.456
only window	0.338	0.363	0.333	0.380	0.479	0.558
only dialogue	0.443	0.454	0.559	0.571	0.726	0.738
no exchange	0.466	0.494	0.584	0.611	0.747	0.764
no window	0.460	0.471	0.578	0.589	0.737	0.747
no dialogue	0.398	0.415	0.457	0.480	0.622	0.643
all	0.475	0.495	0.596	0.616	0.757	0.770

sets having 50 parameters for the latter. In conjunction to previous experiments, the results are computed using a linear Support Vector Machine (SVM) in a 10-fold cross-validation setting. The results are stated in Table 4.13 and visualised in Figure 4.22.

Best performance for both BASE*di* and EXT*di* in terms of UAR, κ, and ρ is achieved by using all parameters. However, it is highly notable that the results are very similar compared to the results of using all but the exchange level parameters (*no exchange*). In fact, applying the Wilcoxon test (Wilcoxon, 1945) for statistical significance proves the difference to be non-significant (BASE*di*: $p = .152$, EXT*di*: $p = .942$). This is underpinned by the results of only using the parameters on the exchange level (*only exchange*) being among the worst performing configurations together with the *no window* results. However, comparing the *all* results to the *no window* results (BASE*di*: $p = .088$, EXT*di*: $p < .0001$) reveals that the window parameters play a bigger role in the overall performance.

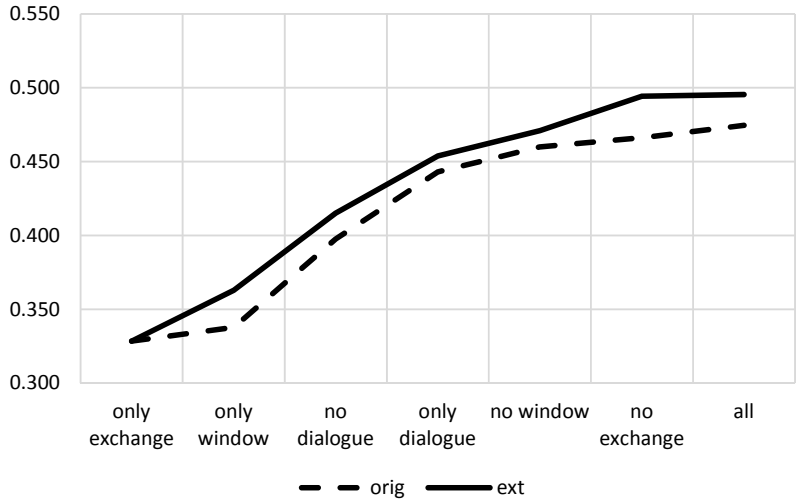


Fig. 4.22: SVM performance in UAR for including and excluding different parameter levels for $LEGO_{orig}$ and for $LEGO_{ext}$.

While the analysis above is true for both feature sets BASE*di* and EXT*di*, the results clearly show that the extension of the feature set results in an increased performance on almost all levels. The overall performance using *all* parameters has been relatively increased by 4.4% ($p < .0001$) in UAR and the performance of *no exchange* has been relatively increased by 6.0% ($p < .0001$). The results of the *only exchange* parameters are the same for both feature sets as the parameters on this level are the same, i.e., have not been computed anew.

While the impact of the different parameter levels on the overall estimation performance is of interest, we are also interested in how the window size influences the estimation performance. Hence, experiments have been conducted with different window sizes. As the experiments above showed that EXT*di* performed significantly better than BASE*di*, only the EXT*di* feature set is used. Again, all experiments are conducted applying 10-fold cross-validation using a linear SVM. The results for UAR, κ, and ρ are depicted in Figure 4.23. Table 4.14 shows also the relative improvement compared to a window size of three used as baseline for this experiment.

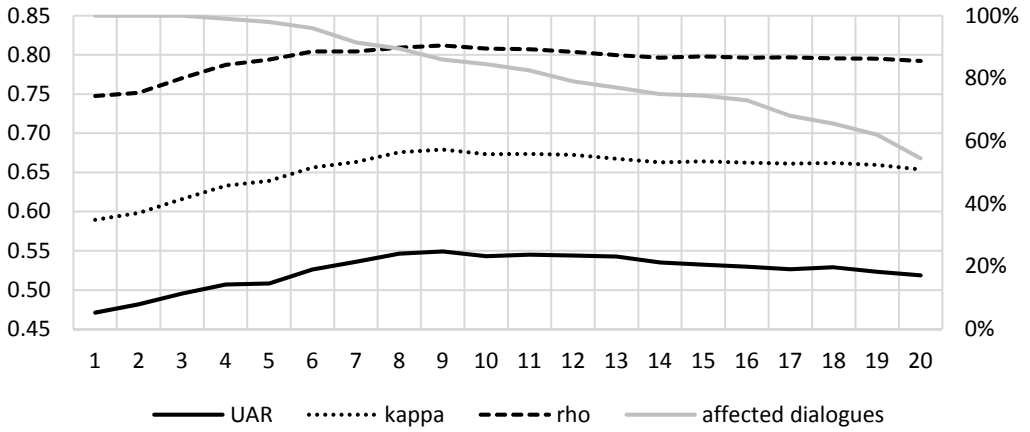


Fig. 4.23: SVM performance (left ordinate) for EXT*di* using different window sizes from $n = 1$ (no window) to $n = 20$ (abscissa). The percentage of affected dialogue, i.e., which have a length greater than the window size, is shown on the right ordinate.

A maximum performance is reached with a window size of 9 for UAR, κ and ρ alike. In fact, an UAR of 0.549 represents a relative improvement compared to a window size of 3 by +10.82%. This clearly shows the potential hidden in these window parameters. If these results are compared to the performance of the original feature set of BASE*di*, the performance is even relatively improved by +15.69%.

It is interesting, though, that the best window size is nine. We believe that this is system dependent and, in Let's go, related to the minimum number of system-user-exchanges necessary to perform a successful dialogue. Looking at the corpus reveals that a minimum of nine exchanges is needed.

4.1.8 Conclusions on User Satisfaction Recognition

Recognising User Satisfaction within spoken dialogue interaction for its usage in adaptive dialogue modelling is not a trivial task and encompasses several problems which we have identified

Table 4.14: Results of different window sizes for IQ recognition in UAR, κ, and ρ. In addition, the relative improvement in UAR with respect to a window size of 3 is depicted. Significance is indicated with * ($p < 0.05$) and ** ($p < 0.01$) determined using the Wilcoxon test (Wilcoxon, 1945). Best performance is achieved for a window size of 9.

Window	UAR		κ	ρ	#dial.
1	0.471	- 4.93%**	0.589	0.747	100%
2	0.482	- 2.76%**	0.598	0.752	100%
3	0.495	–	0.616	0.770	100%
4	0.507	+ 2.30%	0.633	0.787	99%
5	0.508	+ 2.57%**	0.639	0.794	98%
6	0.526	+ 6.16%**	0.656	0.804	96%
7	0.536	+ 8.16%**	0.663	0.804	92%
8	0.546	+ 10.22%**	0.675	0.809	90%
9	0.549	+ 10.82%**	0.679	0.812	86%
10	0.543	+ 9.61%**	0.673	0.808	85%
11	0.545	+ 9.97%**	0.674	0.807	83%
12	0.544	+ 9.76%**	0.672	0.804	79%
13	0.542	+ 9.50%**	0.668	0.800	77%
14	0.535	+ 7.99%*	0.663	0.797	75%
15	0.532	+ 7.42%**	0.664	0.798	75%
16	0.530	+ 6.90%**	0.663	0.796	73%
17	0.526	+ 6.23%**	0.661	0.797	68%
18	0.529	+ 6.75%*	0.662	0.796	66%
19	0.523	+ 5.54%*	0.659	0.795	62%
20	0.519	+ 4.66%*	0.654	0.792	55%

within this section. It already starts with the data acquisition where it is quite difficult to get satisfaction information from real users in a realistic scenario. Using expert raters instead remedies this problem. Based on the Interaction Quality paradigm, we have created our own corpus data which has then been used as subject of multiple estimation approaches. There, we proposed and evaluated novel static and sequential approaches having both groups showing adequate performance. In particular, our best performing approaches have used a plug-in architecture where virtually any static classification approach may be plugged in. Here, we were able to improve the performance of any plugged-in classifier tested. Hence, if new methods for IQ estimation employing a static classifier are found, the expectation is that those approaches may even further be improved by applying our proposed methods.

One important aspect for the analysis is to which degree the approaches are able to *generalise* to similar but still different systems. While for static classification approaches, we identified the rule learner to result in best performance on the available data, is has been found to not generalise well. Here, we have shown that applying a support vector machine provides better generalisation capabilities while still offering reasonable estimation performance. In general, our analysis of the role of the different feature groups in generalisation revealed that the dialogue management parameters are most important. This may most likely be attributed to the fact that those are very general and the most independent form the actual dialogue system.

Another important aspect when analysing the estimation performance is the *contribution of sequential or temporal information*. Here, we analysed several Markovian approaches identifying a significant improvement. Obviously, including information about the previous user satisfaction state into the estimation process improves the overall performance achieving a relative increase in performance of +7.87% compared to the baseline with an UAR of 0.521. However, modelling the temporal dependencies and effects within the parameters used for the estimation is even more promising improving UAR by +15.69% (0.549 UAR). Furthermore, the temporal parameter optimisation achieves also the best performance rates in Cohen's Kappa ($\kappa = 0.679$) and Spearman's Rho ($\rho = 0.812$).

In our experiments, the *temporal information* encoded in the parameters has clearly shown to have a major effect on the estimation performance. In fact, the dialogue level parameters contributing most may be interpreted as the satisfaction of the user mainly depends on the complete dialogue and not on short-term events. However, putting this long-term information in the context of a shorter more recent period modelled by the window level achieves an even better performance. This increase is even more evident when further adjusting the window size. Hence, it may be concluded that the user satisfaction does not solely depend on local effects but those local effects should be interpreted within the context of the complete dialogue.

To sum up, temporal information is very important for recognising the Interaction Quality. This may be represented implicitly within the statistical model or explicitly within the set of features. Depending on the task at hand, both ways of representing the temporal information may be beneficial for the overall recognition performance.

While the major part of this chapter covers our novel approaches for recognising the user satisfaction, within this thesis, we will also propose novel concepts for the recognition of other user states which are also important for adapting the course of the dialogue. Here, the user satisfaction is clearly the most influential one. Still, the *perceived coherence* of a dialogue system is a general concept which thus may be attributed to most systems. This is different to *emotions* or *intoxication*, which both do not occur very often in actual task-oriented human-machine conversations. These effects will, of course, become more visible once the machine has emerged to be a social companion.

Having this in mind, we will continue with presenting the three user states under consideration and describe our work on predicting them in a dialogue context. After presenting all recognition approaches, we will evaluate them on the respective user states following the same structure as in the section on user satisfaction recognition.

4.2 Perceived Coherence Recognition

An important aspect of interaction is the coherence of the dialogue partners' behaviours. For instance, if you talk to someone and ask a question, if the response is not an answer to the question but something totally different, this would be very strange. Hence, this will at least hinder the conversation or even lead to a complete termination. The same is true for human-machine conversations where it is important for the machine to generate coherent responses. Hence, we aim at detecting this coherence automatically. But first, the term *coherence* itself will be elaborated on in the following.

4.2.1 Perceived Coherence

As *coherence* may be a vague concept, we will first give more details on its meaning. It is defined in general by the Longman Dictionary of Contemporary English (Lon, 2003) as "when something [..] is easy to understand because its parts are connected in a clear and reasonable way". Thus, coherence may be seen as an objective concept[13].

Other authors like Perrault et al. (1978) describe coherence in a dialogue context as "whether a sequence of [...] acts were perceived as contributing to the achievement of an overall goal". Here, a subjective aspect is added to the understanding of coherence: the question of how acts are perceived by the user. Within this thesis, the latter aspect is of special interest as we are focusing on how users perceive a system's behaviour. Therefore, we refer to this concept as *perceived coherence*.

Hence, we propose an approach exploiting automatic methods to predict coherence of system dialogue acts on the turn-level. In contrast to previous work, though, we aim at predicting the coherence of the system dialogue act to be executed, i.e., looking into the future. This means that given a dialogue situation, we aim at predicting the coherence of the next system dialogue act scheduled for execution. For this, features both derived from the interaction and encoding the dialogue state will be used. As we focus on perceived coherence, features only representing the dialogue state are not able to model these effects adequately. Cues extracted from the interaction itself, e.g., ASR performance, will be used additionally.

For this, we created an approach which aims at employing methods from static machine learning as presented in Section 2.2 and will be explained in the following.

4.2.2 Statistical Modelling of Recognising the Perceived Coherence of System Dialogue Acts

For automatic recognition of the perceived coherence of system dialogue acts (SDAs), a supervised classification approach is used. Moreover, as already stated, we are interested in predicting the SDA which will be executed next by the SDA. Here, the next SDA is the system act which the strategy represented in the corpus has chosen to execute next. The coherence classes are "strongly coherent", "weakly coherent", and "non-coherent". We assume a relationship between the events of the interaction and the coherence. These events are represented by the automatically derivable interaction parameters as used for the Interaction Quality (see Sec. 4.1.1). Using the interaction parameters along with a representation of the dialogue state called dialogue register as input parameters, a support vector machine (SVM, see 2.2.1) is used to predict coherence.

The overall recognition process is shown in Figure 4.24. A set of interaction parameters, the dialogue state, and the next system dialogue act are used to train an SVM to predict coherence. During evaluation, one single set of interaction parameters, dialogue state and SDA are fed into the statistical model to create a coherence hypothesis.

To evaluate our approach, we rely on the same corpus as for our experiments on Interaction Quality recognition which will be described next.

[13] This corresponds to the understanding of coherence used in previous work on dialogue coherence (cf. Chapter 3)

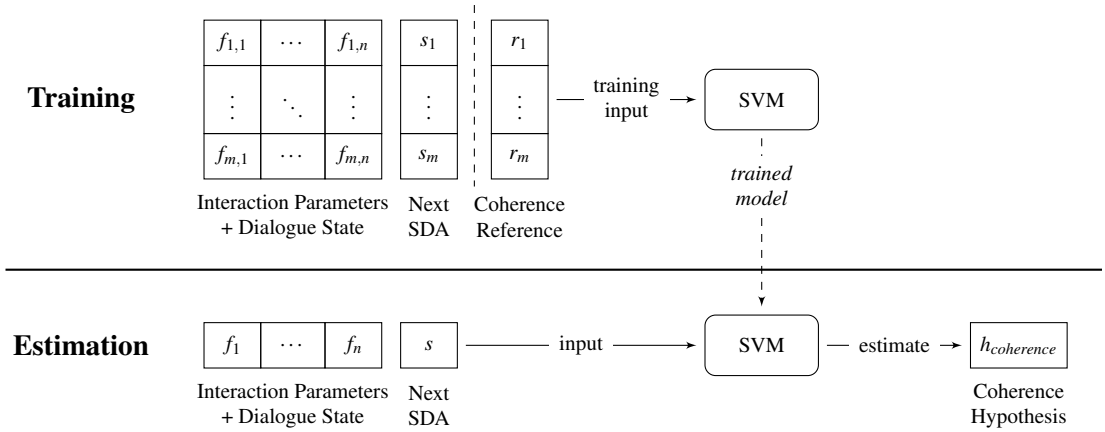


Fig. 4.24: General scheme of a static classifier for coherence estimation: to create a statistical model for coherence estimation (here an SVM), a set of m interaction and dialogue state parameter vectors $f_{m,n}$—consisting of n parameters each—is combined with the next dialogue acts s_m to be analysed. Furthermore, the annotated coherence references r_m are needed. To estimate the hypothesis $h_{coherence}$, the trained model is used to evaluate one input sample.

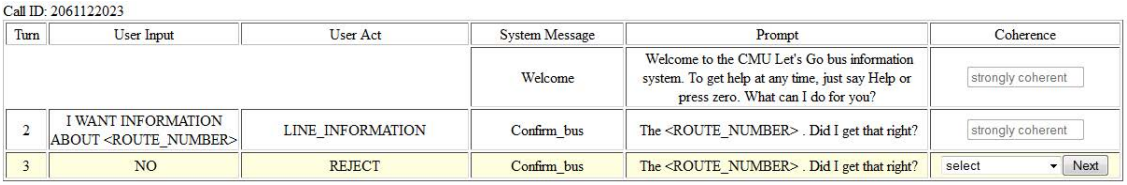
4.2.3 Dialogue Coherence Data

For our experiments, we use the *LEGO* data already presented for IQ estimation in Section 4.1.5. However, for the purpose of perceived coherence estimation, the data needs to be pre-processed. The user dialogue acts (UDAs) have been clustered based on identical dialogue situations to decrease their dimensionality. For instance, *arrival-covered-part*, *arrival-covered-monument*, and *arrival-covered-neighbourhood* are all instances of the general *ARRIVAL* UDA, as they all correspond to situations in which the user provides a valid arrival place. The distinction is only important to query the database. An equivalent process has been applied to System Dialogue Acts. Here, non-semantic concepts have also been discarded as they do not contribute to the progression of the dialogue.

To represent the current dialogue state, a *Dialogue Register* (DR) similar to the one proposed by Griol et al. (2008) is used. The structure encodes the dialogue state in predefined slots containing information about the presence or absence of concepts and attributes (i.e., whether each relevant piece of information has been correctly provided or not). For the *LEGO* corpus, the attributes are *arrival*, *departure*, *busRoute*, and *time* and the concepts are *yes*, *no*, *help*, *repeat*, *start*, and *goodbye*. Each slot may take one of the values "one" or "zero" indicating if the respective concept is currently active or not. This is the same for the attributes. However, attributes may additionally take the value "two" if the attribute value provided to the system has been grounded with the user.

To augment the *LEGO* corpus with coherence information, an expert annotator rated each system dialogue act with a coherence label which is one out of "strongly coherent", "weakly coherent", and "non-coherent". Here, the rater labels if the selected system dialogue act which will be performed next by the system is coherent given the dialogue up to the current turn. In order to do this, the dialogue is presented turn by turn. For example, in Figure 4.25, the rater is deciding about the coherence of the third system turn and has already annotated the first two as

"strongly coherent". Per each turn, information about the user dialogue act (e.g., in turn 2, the user provided a bus line), the dialogue act that the system generated in response (e.g., in turn 2, the system confirmed the bus line), a generic user input, and a generic system message is shown. The generic messages have been added for the raters so that they have a better understanding of what the dialogue acts represent.


Call ID: 2061122023

Turn	User Input	User Act	System Message	Prompt	Coherence
			Welcome	Welcome to the CMU Let's Go bus information system. To get help at any time, just say Help or press zero. What can I do for you?	strongly coherent
2	I WANT INFORMATION ABOUT <ROUTE_NUMBER>	LINE_INFORMATION	Confirm_bus	The <ROUTE_NUMBER> . Did I get that right?	strongly coherent
3	NO	REJECT	Confirm_bus	The <ROUTE_NUMBER> . Did I get that right?	select Next

Fig. 4.25: The coherence labelling web form showing user acts and system messages along with generic user inputs and system prompts to help the rater interpret the acts. The exchange to be labelled is highlighted in light yellow.

To generate perceived coherence, it is very important for the labelling process that the rater *assumes* the position of the system. That means, the question the rater asks himself each time applying a rating is: "Given the accumulated information about the user's intention and what has been said in the dialogue by the user and the system so far: does it make sense for me (the system) to perform this dialogue act now?". The first SDA "Welcome" is always labelled as "strongly coherent".

The coherence label distribution of the labels applied to the *LEGO* corpus are presented in Table 4.15. The vast majority has been labelled as coherent with only less than 10% being labelled as non-coherent.

Based on this pre-processed corpus, our approach on predicting the coherence of the next system action is evaluated in the next section.

Table 4.15: Distribution of coherence labels within the *LEGO* corpus in absolute amount and percentage with respect to all exchanges.

strongly coherent	weakly coherent	non-coherent
2822 (66.8%)	1005 (23.8%)	395 (9.4%)

4.2.4 Evaluation of Perceived Coherence Recognition

For predicting the coherence, we relate the coherence to parameters of the interaction. As these interaction parameters are derived from three different SDS modules, i.e., ASR, SLU, and DM, the contribution of each of these modules to the overall performance is analysed as well as the role of the dialogue register. Hence, several feature configurations are applied.

The basic feature set *BASE* contains three features: the exchange number, the previous dialogue act and the next dialogue act. These features represent the minimal information necessary to

predict coherence of the next SDA. Thus, they are also part of all other feature sets. To investigate the contribution of the interaction parameters, the feature sets *ASR*, *SLU*, *DM*, and *IP* are created. The former three contain the interaction parameters (on all three levels) which are derived from the respective SDS module while the latter combines all interaction parameters. Furthermore, the contribution of the dialogue register is analysed in the *REG* feature set. The performance using all parameters is represented in the *IP+REG* feature set.

The experiments are conducted employing a SVM with linear kernel in a 10-fold cross-validation setting. The performance is measured using the unweighted average recall (UAR). UAR is derived from the class-wise recalls by computing the arithmetic average.

The results are depicted in Table 4.16. As can be seen, all feature groups performed better than the majority baseline of 0.333[14]. Using all features (*IP+REG*) achieved the best result of 0.623 UAR. This shows the viability of this approach. However, while it may be assumed that the dialogue register, i.e., the encoded dialogue state, highly contributes to the overall performance, it is quite the opposite: *REG* performs worst with an UAR of 0.366. Most surprisingly, the performance is even worse than only using the *BASE* features. This may be attributed to the uncertainty added by the number of features of the *REG* group compared to the information contained. Regardless, this information is still beneficial as it increases the performance of the *IP* feature group by 0.014. The most information, though, is encoded in the *IP* features with an UAR of 0.609. Here, the *DM* and *ASR* features contribute most.

Table 4.16: The results in UAR of SDA coherence prediction for different feature groups applying a linear SVM.

	UAR	# features
BASE	0.384	3
ASR	0.514	29
SLU	0.400	8
DM	0.587	18
IP	0.609	49
REG	0.366	13
IP+REG	0.623	59

4.2.5 Conclusion on Perceived Coherence Recongition

We have proposed and evaluated an approach on predicting the perceived coherence of the next system dialogue acts. This approach may be utilised within the dialogue management module for influencing the selection of the next system dialogue act. As it is based on an abstract representation comprising dialogue acts and interaction parameters, it is generalizable to different application domains.

Using interaction parameters and the encoded system state as input parameters to a statistical classifier results in an UAR of 0.623. This may be considered as good performance given the

[14] Majority baseline means that always the majority class is assigned. For 3-class problems, this always results in an UAR of 0.333.

complexity of the task. Furthermore, we analysed the contribution of different feature sub-groups showing that interaction parameters derived from the dialogue management contribute most.

However, there are still some shortcomings which should be addressed in future work. First, the data is highly unbalanced. Sampling methods or introducing a cost matrix may compensate for that. Furthermore, as we intend to predict the coherence of the next system dialogue act given a dialogue situation, the corpus should be extended by more SDAs given one dialogue situation.

Deriving the perceived coherence clearly may help to change the course of the dialogue in a straight-forward way, this is different for emotions. Still, for rendering a system as social competent, emotions should be taken account of nonetheless. This will be described in the following.

4.3 Emotion Recognition

Recognising the current emotional state of the user is a hard task. And while emotions may occur rarely in human-machine conversations, technical systems of the future should be equipped with the capability of understanding emotions in order to be regarded as social competent. For example, the system may apply strategies aiming at reducing the users' anger level. For this, the emotional state of the user needs to be identified or estimated, respectively.

In human-human communication, people are usually quite capable of identifying the emotional state of the other person, while machines still do have a hard time recognising people's emotions with the same accuracy. State-of-the-art approaches for automatic emotion recognition regard the problem independently of the speaker. However, while the basic emotions are shared between all people and cultures (Scherer, 2002), humans have a fine-tuned emotional model of people they know allowing for recognising their emotions more accurately. Furthermore, speaker-specific models have shown to improve speech recognition as well (e.g., (Leggetter and Woodland, 1995)). Hence, we add speaker-specific information to the emotion recognition process and evaluate if this may result in an increase performance in general (Sidorov et al., 2014b).

Consequently, we will first present two approaches on speaker-dependent emotion recognition. Here, we will focus on classification-based emotion recognition (opposed to estimating points the PAD space, cf. Sec. 3.1.3). After describing the data we use for our tests, the evaluation procedure and the results will be described in detail.

4.3.1 Speaker-dependent Emotion Recognition

Incorporating speaker-specific information into the emotion recognition process may be done in many ways. Within this thesis, we propose two ways: to add the speaker information to the set of features and to create speaker-dependent models. While, for conventional emotion recognition, one statistical model is created independently of the speaker, one may create separate emotion model for each speaker. Both approaches result in two-stage recognition procedures (see Figure 4.26 and Figure 4.27): first, the speaker is identified. Then, this information is included into the feature set directly by extending the feature vector resulting in one *combined* emotion model (Feature Set Extension, FSE), or one out of several *individual* emotion models is selected depending on the speaker (individual models, IM). Both hybrid systems for emotion recognition and speaker identification (ER-SI) have been investigated and evaluated in this study.

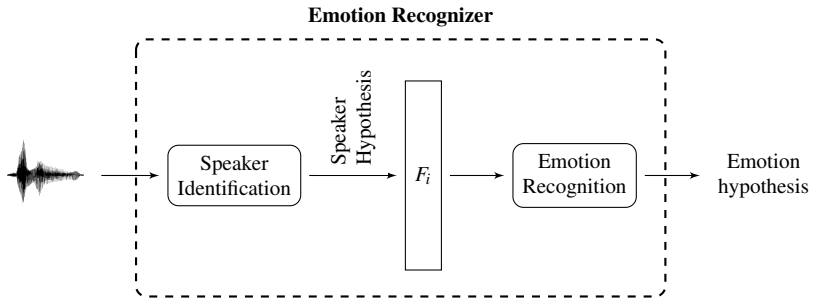


Fig. 4.26: Hybrid two-stage emotion recognition system FSE originally published in (Sidorov et al., 2014b): the speaker information is added to the feature set resulting in one combined emotion model for all speakers.

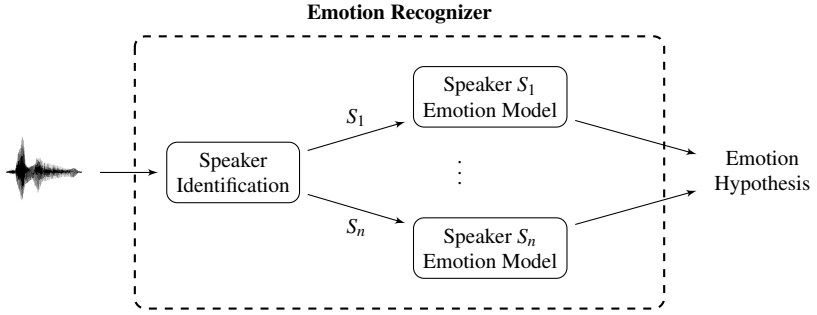


Fig. 4.27: Hybrid two-stage emotion recognition system IM originally published in (Sidorov et al., 2014b): an individual emotion model for each speaker is used selected based on the speaker estimation.

4.3.2 Emotion Data

For testing the two proposed approaches on speaker-dependent emotion recognition, a number of speech databases have been used which will be described in the following. These different data sets were chosen as they offer a variety of different characteristics including database language, acted vs. non-acted speech, and the number of emotions.

Berlin The Berlin emotional database (Burkhardt et al., 2005) was recorded at the Technical University of Berlin and consists of labelled emotional German utterances which were spoken by 10 actors (5 female). Each utterance has one of the following emotional labels: neutral, anger, fear, joy, sadness, boredom, and disgust.

Let's Go The Let's Go database (see Sec. 4.1.5) does not only contain information about the user satisfaction but also comprises emotion labels for the emotions angry, slightly angry, very angry, neutral, friendly, and non-speech (critical noisy recordings or just silence). The database contains non-acted American English utterances extracted from an automated bus information system of the Carnegie Mellon University in Pittsburgh, USA.

SAVEE Haq and Jackson (Haq and Jackson, 2010) recorded the SAVEE (Surrey Audio-Visual Expressed Emotion) corpus for research on audio-visual emotion classification from four native English male speakers. Each acted utterance has been labeled with an emo-

tion from the standard set of emotions (anger, disgust, fear, happiness, sadness, surprise, and neutral).

UUDB The UUDB (The Utsunomiya University Spoken Dialogue Database for Paralinguistic Information Studies) database (Mori et al., 2011) consists of spontaneous Japanese human-human speech. Task-oriented dialogue produced by seven pairs of speakers (12 female speakers) resulted in 4,737 utterances in total. Emotional labels for each utterance were created by three annotators on a five-dimensional emotional basis (interest, credibility, dominance, arousal, and pleasantness). For this work, only pleasantness (or evaluation) and the arousal axis are used. The corresponding quadrant (counterclockwise, starting in positive quadrant, assuming arousal as abscissa) are then assigned to emotional labels: happy-exciting, angry-anxious, sad-bored and relaxed-serene (Schuller et al., 2009b).

VAM Based on the popular German TV talk-show "Vera am Mittag" (Vera in the afternoon), the VAM-Audio database (Grimm et al., 2008) has been created at Karlsruhe Institute of Technology. The emotional labels of the first part of the corpus (speakers 1–19) have been labelled by 17 human annotators. The remaining utterances (speakers 20–47) were labelled by six annotators. All emotional labels are based on the three-dimensional PAD emotion space. The emotional labelling was performed in a similar way to the UUDB corpora, using pleasure and arousal axis.

While the Berlin and SAVEE corpora consist of acted emotions, the other three databases comprise real emotions. Furthermore, for the two languages English and German, acted and non-acted emotions have been considered. Only for Japanese, solely non-acted emotions were available. A statistical description of the used corpora may be found in Table 4.17. It should also be noted that Let's Go, UUDB, and VAM are highly unbalanced.

Table 4.17: Emotion database descriptions used for speaker-dependent emotion recognition originally published in (Sidorov et al., 2014b).

Database	Language	length	# emotions	Notes
Berlin	German	24.7 min	7	Acted, single utterances
Lets Go	English	118.2 min	5	Non-acted, human-machine
SAVEE	English	30.7 min	7	Acted, single utterances
UUDB	Japanese	113.4 min	4	Non-acted, human-human
VAM	German	47.8 min	4	Non-acted, human-human

Emotions themselves as well as annotating emotions both have a subjective nature. That is why it is important to have several people who apply the emotional labels. Even for humans, it is not always evident to make a decision about a detected emotion. Each study which proposed an emotional database provides also a confusion matrix and statistical description of the decisions of the human annotators.

The presented databases having different difficulty levels are used to test and evaluate our proposed speaker-dependent approaches. The evaluation process and the results will be described in the next section.

4.3.3 Evaluation of Speaker-dependent Emotion Recognition

For evaluation of the proposed speech-based emotion recognition approaches, the choice of a set of appropriate speech signal features is still an open question, both for speaker identification and for emotion recognition. As the focus of this study lies on improving emotion recognition by adding speaker dependency (and not on tweaking the feature set to the max), no feature set optimisation has been applied and simply the most popular features have been chosen (cf. (Schmitt et al., 2009b)). Hence, the feature vector includes average values of the following speech signal features: power, mean, root mean square, jitter, shimmer, 12 MFCCs, and five formants. Mean, minimum, maximum, range, and deviation of the following features have also been used: pitch, intensity and harmonicity. This results in a 37-dimensional feature vector for one speech signal file. To extract these speech signal features from the audio stream, the Praat system (Boersma, 2002) was used.

For each of the two proposed speaker-dependent emotion recognition approaches (FSE and IM), the according statistical models have been used to detect emotions, or the speaker, respectively, in a static mode. This means that each speech signal was parametrised by one single 37-dimensional feature vector consisting of the corresponding average values. As this study is focused on the theoretical improvement of emotion recognition using speaker-specific information, the usage of other speech signal features or modelling algorithms may improve the recognition performance even further.

To investigate the *theoretical* improvement of using speaker-specific information for ER, the true information about the speaker has been used. Then, in order to provide pilot experiments, a real SI component has been applied. For both tasks (ER and SI), a multi-layer perceptron (see Sec. 2.2.1), which is a baseline type of artificial neural networks (ANN), has been chosen as a modelling algorithm for both speaker identification and emotion recognition.

As a baseline, an emotion recognition process without speaker-specific information has been conducted. Dividing the data into training and test sets, the training set was used to create and train the ANN-based emotion model. The test set was used to evaluate the model. Hence, one single neural network has been created addressing the emotions of every speaker in the database.

In the first experiment, the focus was on investigating the theoretical improvement which may be achieved by using speaker based adaptivity. For this, known speaker information (true labels) was used for both approaches (FSE and IM). For FSE, the speaker information was simply added to the feature vector. Hence, all utterances with the corresponding speaker information were used to create and evaluate an ANN based emotion model. For IM, individual emotion models were built for each speaker. During the training phase, all speaker utterances were used for creating one emotion model for each speaker. During testing, all speaker utterances were evaluated with the corresponding emotion model.

Additionally, a second experiment was conducted including a real speaker identification module instead of using the ground truth speaker information. First, an ANN-based speaker identifier was created during the training phase. Furthermore, for FSE, the known speaker information was included into the feature vector for the training of the emotion classifier. The testing phase started with the speaker identification procedure. Then, the speaker hypothesis was included into the feature set which was in turn fed into the emotion recogniser. For IM, an ANN-based emotion recogniser was created for each speaker separately. For evaluating this system, the speaker hypothesis of the speaker recognition is used to select the emotion model which corresponds to

Table 4.18: Evaluation Result of FSE in percent originally published in (Sidorov et al., 2014b): Accuracy of baseline (Without SI), Experiment 1 (True SI) and Experiment 2 (ANN SI, having SI accuracy in parentheses.)

Database	Without SI	True SI	ANN SI
SAVEE	60.39	62.64	63.58 (99.06)
Berlin	73.76	77.74	74.61 (73.24)
VAM	68.73	71.48	69.00 (66.39)
UUDB	89.99	90.42	90.10 (72.85)
Let's Go	76.37	78.95	77.50 (42.41)

Table 4.19: Evaluation Result of IM in percent originally published in (Sidorov et al., 2014b): Accuracy of baseline (Without SI), Experiment 1 (True SI) and Experiment 2 (ANN SI, having SI accuracy in parentheses.)

Database	Without SI	True SI	ANN SI
SAVEE	60.39	66.08	65.69 (99.14)
Berlin	73.76	74.01	68.84 (74.06)
VAM	68.73	67.76	64.80 (66.73)
UUDB	89.99	89.91	89.46 (73.47)
Let's Go	76.37	80.96	75.47 (44.03)

the recognised speaker to create an emotion hypothesis. In contrast to the first experiment, these experiments using estimated speaker information are not free of speaker identification errors.

In order to generate more statistically significant results, the complete classification process was run 25 times for each database and experiment. For each run, the databases were randomly divided into training and test sets (70–30% correspondingly). While each database was stratified into training and test sets by the *emotion* class, the Let's Go database was stratified into subsets by the *speaker* class, due to highly unbalanced distribution of the speaker class (IM only). For all experiments, z-score normalisation has been applied for all features. The final results are shown in Table 4.18 for FSE, where speaker-specific information is included into the feature set, and in Table 4.19 for IM, where separate emotion models are created for each speaker. The results are calculated taking the average of all runs. The first columns show the baseline of ANN-based emotion recognition accuracy (without speaker-specific information). In the second column, the accuracy of the emotion recognition systems using known speaker information is presented followed by the emotion recognition accuracy based on ANN-based speaker identification. For the latter, values within the parentheses depict the performance of the speaker identification module.

As a result, we can conclude that the addition of speaker-specific information in the emotion recognition procedure may significantly improve the recognition performance. For all corpora, the recognition accuracy has been improved by adding speaker information to the feature vector (see Table 4.20). This improvement is even significant for almost all databases.

Table 4.20: Improvement in ER performance using the true speaker information (True SI) and using SI information of an ANN (ANN SI) originally published in (Sidorov et al., 2014b). Significant differences are marked with ** ($\alpha < .05$) and * ($\alpha < .01$) using the T-test.

Database	SI in FS (Sys. A)		Separate Models (Sys. B)	
	True SI	ANN SI	True SI	ANN SI
SAVEE	+3.72% *	+5.28% **	+9.42% **	+8.77% **
Berlin	+5.39% **	+1.15%	+0.33%	-6.67% **
VAM	+4.00% **	+0.39%	-1.41%	-5.71% **
UUDB	+0.47% **	+0.12%	-0.08%	-0.58% **
Let's Go	+3.37% **	+1.47% **	+6.01% **	-1.17% **

4.3.4 Conclusion on Speaker-dependent Emotion Recognition

It is evident that already a very simple method as extending the feature vector with additional speaker-specific information may improve the ER accuracy. This has been shown for five different databases even if using a real SI module. This improvement is significant when using true SI information for most of the used corpora (see Table 4.17). These results are very encouraging leading to further more sophisticated approaches on speaker-dependent emotion recognition, e.g., applying methods known from speaker-dependent speech recognition.

However, for building accurate individual emotion models, balanced databases are required. In order to build one emotion model for each speaker, a high number of emotional samples per speaker are needed. Hence, for some of the corpora (VAM and UUDB), the addition of speaker information (as a building of separate emotion models) could not improve the recognition accuracy.

Moreover, decomposing a problem as proposed by us with the separation of speaker identification and emotion recognition favours the accumulation of errors as can easily be seen by the complete probability formula for the probability $P(em)$ of emotion *em*

$$P(em) = P(sp) \cdot P(em|sp) + P(\overline{sp}) \cdot P(em|\overline{sp}) \; . \tag{4.21}$$

Here, $\mathtt{P}(sp)$ denotes the probabilities of recognising the correct speaker and $\mathtt{P}(\overline{sp})$ is the probability of recognising the wrong speaker. Thus, $\mathtt{P}(em|sp)$ is a conditional probability for emotion *em* given the correct speaker and $\mathtt{P}(em|\overline{sp})$ is the probability for emotion *em* given the false speaker. In other words, estimating the emotion correctly may also happen when estimating the speaker wrongly but still estimating the emotion correctly. This corresponds to the similarity of emotions between speakers. Even if a high value of SI accuracy as 99.14% is achieved, there is still a gap between emotion recognition using known speaker information (+9.42%) and an actual SI module (+8.77%) (see Table 4.20).

While an ANN already provides reasonable results for speaker identification, we are still examining its general appropriateness. The usage of other—possibly more accurate—classification approaches may improve the performance of our proposed hybrid system. Furthermore, dialogues do not only consist of speech, but also of a visual representation. Hence, an analysis of picture or even video recordings may also improve SI and ER performance.

After introducing emotions as a speech-based user state which the dialogue may adapt to, a more specialised user state may also be considered when designing dialogue systems: the intoxication level. Its estimation process is similar to the process of recognising emotions. This along with a comparative user study will be presented in the next section.

4.4 Intoxication Recognition

Another aspect of the user state is intoxication, i.e., the degree by which the user is drunk. This may be measured in blood alcohol concentration (BAC). The intoxication information may be used within a spoken dialogue system, e.g., to deny specific services if the user is identified to be drunk. Hence, within this section, we present work on automatically estimating the intoxication level of users (Ultes et al., 2011b). As this is a hard task even for humans, we further perform a study on humans to answer the question how well humans are able to identify if someone is drunk. This is especially difficult as only the recorded speech of the users may be used for this task (for both, humans and machine).

4.4.1 Statistical Classification

For creating a statistical classification approach to automatically classify intoxication, we will employ a support vector machine—a standard supervised classification approach—for discriminating between "intoxicated" and "non-intoxicated" rendering the problem as a binary classification task. As input variables, we will use 4368 acoustic and prosodic features extracted from the speech signal. Additionally, we extracted 19 features from the linguistic transcriptions. These features included the total number of words per utterance plus number and rate of repetitions, hesitations, interruptions, corrections, word lengthenings, wrongly pronounced words, and pauses (additionally split into short and long pauses). The overall estimation process is presented in Figure 4.28.

The data which is used for training and evaluation of our approach will be described in the following.

4.4.2 Intoxication Data

The data used for our experiments on intoxication recognition is based on the alcohol language corpus (ACL) (Schiel et al., 2008) of the 2011 Interspeech Speaker State Challenge (Schuller et al., 2011). The speech data contains recordings of speakers who read out loud a sequence of utterances both while being sober and while being drunk. To get drunk, the participants were under constant medical supervision. They had chosen their target intoxication level (Blood Alcohol Concentration, BAC) and according to their physical condition, the medical team calculated the amount of alcohol they needed to drink in order to achieve the target BAC.

For the challenge, the data was divided into three sets: a training set, a development set, and a test set. While there were labels provided for the first two sets, the test set was used to create the challenge performance measure[15]. As the audio samples were annotated with a concrete BAC,

[15] The classification results could be uploaded to a web script which automatically computed the recognition performance.

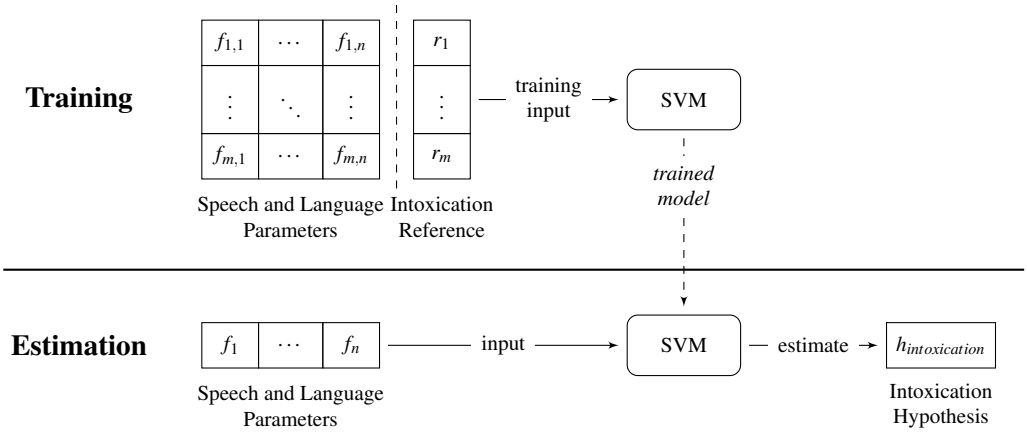


Fig. 4.28: General scheme of a static classifier for intoxication estimation: to create a statistical model for intoxication estimation (here an SVM), a set of m speech and language parameter vectors $f_{m,n}$—consisting of n parameters each—along with their annotated intoxication references r_m are used. To estimate the hypothesis $h_{intoxication}$, the trained model is used to evaluate one input sample.

we labelled all samples belonging to the class "intoxicated" with a BAC > 0. As the resulting set was not balanced regarding "intoxicated" and "non-intoxicated" samples, the data set including the linguistic features was balanced using the Synthetic Minority Over-sampling Technique (SMOTE) (Chawla et al., 2002).

This data set was then used for evaluating of our classification approach. The results of the evaluation will be descried in the next section.

4.4.3 Evaluation of Intoxication Recognition

In order to evaluate our statistical model for recognising intoxication, a linear SVM was used (cf. Sec. 2.2.1). As stated above, the speaker state challenge (Schuller et al., 2011) provided a training set, a development set and a test set. Thus, two configurations were considered: the training set was used for training having the development set used for evaluation and the training set was combined with the development set for training having the testing set for evaluation. Having these configurations, we ensured that no training data was part of the evaluation data. For applying the SVM, the complexity parameter was heuristically optimised resulting in $C = 0.05$. Using this configuration, the SVM achieved an UAR of 0.653. By comparing the performance with the majority baseline of an UAR of 0.5 for a binary classification task, it is evident that the problem of intoxication recognition is not easy to solve even when using 4387 acoustic and linguistic parameters.

To improve the recognition performance, one way is to perform feature set optimisation by identifying and selecting the most meaningful features. Hence, in a second step, feature selection was performed on the training data. For this, all features were weighted using the information-gain ratio (IGR). Through subsequent addition of the highest weighted feature, a learning curve was generated. The current feature set was evaluated using the before mentioned setup using a SVM with the optimal complexity of 0.05. This was done for the training data with the baseline

feature set and the extended feature set alike. However, for the baseline, the complexity was 0.01 (cf. (Schuller et al., 2011)). The UAR was calculated after each evaluation and the resulting curves can be seen in Figure 4.29. It is visible that classification using the extended feature set clearly outperforms classification with the baseline feature set. The extended set yields a maximum UAR of 0.661 with 3015 features versus a maximum UAR of 0.653 with 3839 features for the baseline feature set.

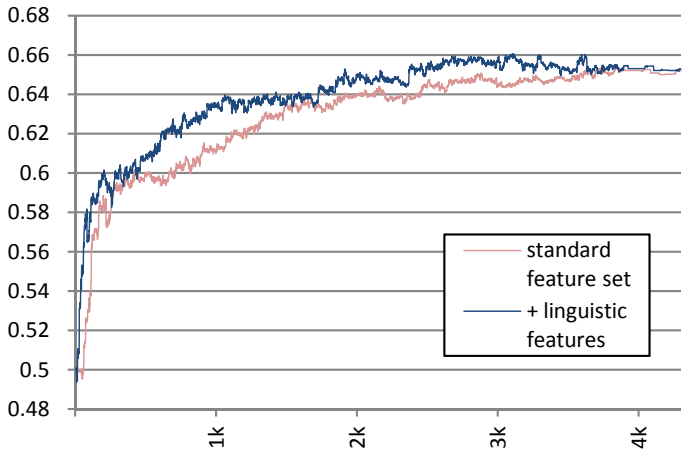


Fig. 4.29: Learning curve of feature selection using IGR originally published in (Ultes et al., 2011b): learning using only acoustic and prosodic features (baseline feature set, red curve) is clearly outperformed by using additional linguistic features (blue curve).

An overview over the results is presented in Table 4.21. Adding linguistic features alone does not yield higher performance. Only with additional ranking using IGR and subsequent feature selection, the performance increases significantly. Using the development and training data together for training of the SVM with the identified optimal configuration (extended feature set, best 3105 features, C=0.05), the evaluation on the test set shows an improvement of the recognition rate of 0.7 percentage points.

In addition, we also applied another feature selection technique also based on IGR ranking. Only features contributing to an increase of the learning curve were used. For that, the UAR ua_i was compared with the UAR ua_{i+1} on the same feature set but with addition of feature $i+1$. Using that, we created two different feature sets: feature set A consisted of all features which fulfilled $ua_{i+1} > ua_i$. Set B consisted of all of the features of set A plus all features, which fulfilled $ua_{i+1} = ua_i$ if and only if the weights of the features $i+1$ and i were equal. These sets were further divided into two instances: instance *all* consisted of the features as described above. Instance *opt* consisted only of these features up to the optimal number of features according to IGR ranking. The UARs are presented in Table 4.22. It can be seen that even a feature set consisting of only 32% of the initial features (acoustic, prosodic, and linguistic) can compete against the full feature set.

As we have presented results on intoxication classification for machines, the question remains how humans are able to perform the same task. To figure this out, a human study has been performed which will be presented in the following.

Table 4.21: Results of intoxication classification originally published in (Ultes et al., 2011b): feature selection on the combined feature set of acoustic, prosodic, and linguistic features using IGR outperforms classification without feature selection or without linguistic features.

Feature set	**# features**	**UAR**
	train vs. devel	
acoustic	4368	0.653
acoustic + linguistic	4389	0.653
acoustic (IGR)	3839	0.653
acoustic + linguistic (IGR)	3105	0.661
	train + devel vs. test	
acoustic (challenge baseline)	4368	**0.659**
acoustic + linguistic (IGR)	3105	**0.666**

Table 4.22: Unweighted average recall of different feature sets based on IGR ranking and the resulting learning curve originally published in (Ultes et al., 2011b).

Feature set	**# features**	**UAR**
A.*all*	2631	0.656
A.*opt*	1726	0.642
B.*all*	1714	0.653
B.*opt*	1392	0.654

4.4.4 Web-based Human Performance Study

Exploiting the alcohol language corpus (ACL) (Schiel et al., 2008) of the 2011 Interspeech Speaker State Challenge (Schuller et al., 2011), a balanced set of 3200 audio samples was selected randomly from the training set. Based on the assumption that the human raters' attention decreases the more samples they rate, the set was further divided randomly into subsets consisting of only 50 audio samples each, whereat one subset was used three times in order to compensate for different rating behaviour. Each rater was asked to assign each sample to one out of the two categories *drunk* and *sober*. The samples were presented to the raters on a web-page presented in Figure 4.30, which also allowed for easy distribution. By that, we were able to reach many raters on short notice. Moreover, to motivate the participants we designed the study as a challenge. The aim was to tell better than the other participants, whether a sample originates from an intoxicated or non-intoxicated speaker. The four best participants were rewarded with book vouchers from 10-20€.

In total there were 79 participants forming a heterogeneous group with an average age of 34.2 years and a standard deviation of 12.6 years. The oldest person was 65 years old and the youngest person was 20 years of age. Out of the 79 people, 48 were male and 31 female.

For the data, the accuracy (ACC) and unweighted average recall (UAR) (both, see Sec. 2.2.3) were applied as measure of performance. These results are presented in Table 4.23. Table 4.24

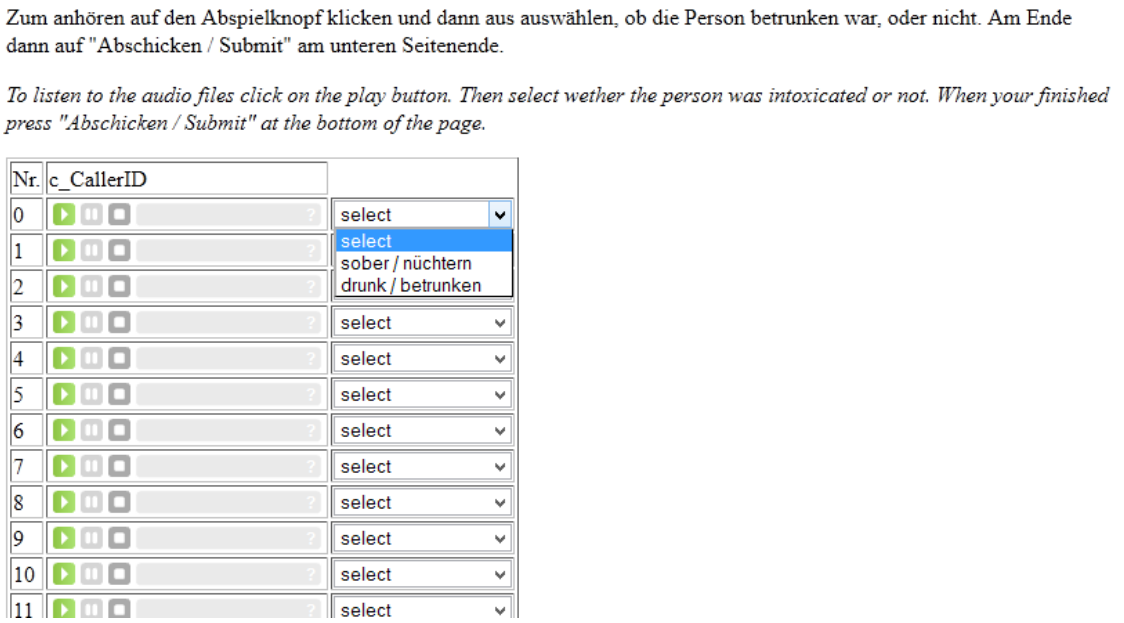


Fig. 4.30: The website for collecting the human ratings for the ALC originally published in (Ultes et al., 2011b).

displays recall and precision of the different classes. It can be seen that the results are independent from gender or age of the raters. Furthermore, the raters yielded a higher precision on recognising intoxicated people but a lower recall on non-intoxicated ones.

Table 4.23: Statistics on the raters originally published in (Ultes et al., 2011b): No obvious distinction can be made about the rater performance with respect to gender or age.

Category	**# participants**	**ACC**	**UAR**
total	79	0.555	0.558
male	48	0.564	0.566
female	31	0.542	0.546
age $<$ 50	65	0.554	0.556
age $\geq$ 50	14	0.561	0.568

Furthermore, not only the performance of the raters on the whole data set was analysed but also subdivisions regarding the blood alcohol concentration (BAC) and the audio sample length. With this, two assumptions should be validated: as, in general, the ability to speak flawlessly decreases with rising BAC, people can tell better that a person is intoxicated. As an implication, the highest uncertainty about the speaker state exists with a low but not zero BAC. Secondly, the longer people listen to speech, the easier they can tell if this person is intoxicated. Thus, UAR is supposed to rise with increasing segment length. The first assumption could be validated as can

Table 4.24: Recall and Precision of the raters originally published in (Ultes et al., 2011b): Recall for non-intoxicated people was better than for intoxicated, whereas intoxicated people generated a higher precision.

Category	intoxicated	non-intoxicated
	recall	
total	0.452	0.664
male	0.461	0.671
female	0.438	0.655
age < 50	0.460	0.652
age ≥ 50	0.414	0.722
	precision	
total	0.587	0.534
male	0.592	0.546
female	0.580	0.517
age < 50	0.581	0.535
age ≥ 50	0.619	0.531

be seen in Figure 4.31. The curve shows that the accuracy for non-alcoholised speech samples is high. It declines rapidly for low blood alcohol concentrations and rises again with rising BAC. (As we only had very little data for 0.2, 0.3 and 1.5 per mill BAC in the corpus, these values can be seen as outliers.)

As for the second assumption, we were not able to validate it. The curve in Figure 4.32 shows no sign of regularity. This indicates that there is no correlation between human rater performance and the duration of the audio material when classifying persons as intoxicated. It should be noted, though, that most of the audio samples had a length less than 10 seconds. For longer sound file durations definite assumptions about the UAR of human raters cannot be made.

Furthermore, we applied majority voting on the ratings. Since each audio sample was rated by three raters and marked as intoxicated or non-intoxicated speech this could yield a clearer vote. Table 4.25 shows the results. The UAR improved only slightly by 1.6 percentage points, which is a first indication that the inter-rater reliability is low.

Table 4.25: Rater performance originally published in (Ultes et al., 2011b): Majority voting slightly improves the overall UAR.

Category	**UAR**
all	0.558
best rater	0.700
worst rater	0.383
maj. voting	0.574

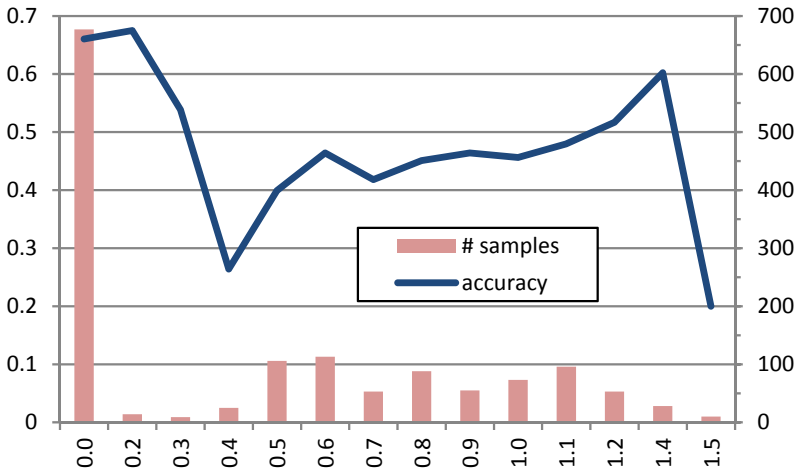


Fig. 4.31: Rater performance with respect to BAC originally published in (Ultes et al., 2011b): Non-alcoholised and strongly alcoholised people are recognised best. Recognition of slightly alcoholised people is harder. The blue curve describes the accuracy (left axis) and the red bars the amount of examples (right axis). The labels on the horizontal axis denote the BAC per mill. So, for instance, out of the 677 samples with a BAC of 0.0 per mill, 66% have been classified correctly. Results for 0.2, 0.3 and 1.5 per mill can be seen as outliers as only limited data was available.

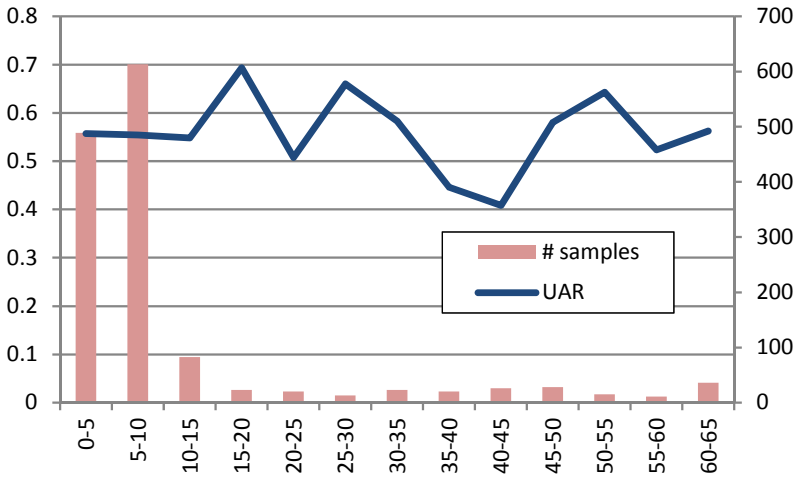


Fig. 4.32: Rater performance with respect to audio sample duration originally published in (Ultes et al., 2011b): There is no visible sign of any correlation between the duration of the audio signal and the ability to tell if the person was intoxicated or not. The blue curve describes the unweighted average recall (UAR, left axis) and the red bars the amount of examples (right axis). The labels on the horizontal axis denote the file duration in seconds of the files assigned to this bin.

The inter-rater reliability was investigated using Cohen's Kappa (Section 2.2.3. It was calculated for each subset by averaging over the Kappas of each rater pair. The resulting Kappa values were further averaged creating the final Kappa value of this study. As can be seen in Table 4.26, the ratings are very inconsistent with $\kappa = 0.15$. It is notably much smaller than Kappa values of other tasks, like recognition of emotion or age, indicating that even humans have a hard time telling if a person is intoxicated or not. (It should be noted that the BAC range of the intoxicated group started at 0.2 per mill.)

Table 4.26: Rater agreement originally published in (Ultes et al., 2011b): classification of intoxication is a hard task for humans and the results majorly depend on the raters. This is further emphasised by comparison with Kappa values of other tasks (values taken from (Shafran et al., 2003) and (Schmitt et al., 2010b)).

Task	**avg. κ**	**max. κ**	**min. κ**
	Alcohol Language Corpus		
intoxication	0.15	0.4	-0.5
	How May I Help You		
emotion	0.42	-	-
age	0.5	-	-
dialect	0.58	-	-
gender	1.0	-	-
	SpeechCycle Broadband Agent		
age	0.21	-	-
anger	0.7	-	-

We further compared the results of the statistical classification with the results of the human raters displayed in Table 4.27. For both human and statistical classification, the recall of non-intoxicated persons is higher than of intoxicated ones. Here, classifier and humans correspond. In contrast to this the SVM recognises non-intoxicated persons better while humans are able to recognise intoxicated persons better. Moreover, statistical classification outperforms human performance considerably. The UAR of humans is 10.8 percentage points below the results of the statistical classifier.

4.4.5 Conclusion on Intoxication Recognition

In this work, we investigated the complexity of the task of distinguishing between intoxicated and sober people merely by speech. A study with human raters has been presented showing that even humans do not agree with each other about whether a person is sober or drunk. This is indicated by very low inter-rater agreement. Furthermore, we implemented a statistical classification mechanism incorporating linguistic features and applying feature selection using IGR ranking and comparing it to IGR ranking performed on the baseline feature set. We showed that only with additional linguistic features better classification results may be achieved using IGR ranking. Finally, by comparing the statistical classifier with human performance, it can be noted that the

Table 4.27: Intoxication recognition originally published in (Ultes et al., 2011b): recall and precision of human raters compared to statistical classification. Whereas recall of non-intoxicated persons is higher than of intoxicated persons for both humans and the SVM, humans have a higher precision on intoxicated and the SVM on non-intoxicated persons.

Classification	**intoxicated**	**non-intox.**	**average**
		recall	
human	0.452	0.664	0.558
statistical	0.625	0.707	0.666
		precision	
human	0.587	0.534	0.561
statistical	0.645	0.689	0.667

statistical classifier yields better results than humans. However, it should be kept in mind that the classifier has been trained with domain data before, whereas the participants have not seen specific samples of the corpus before rating. Since we anticipate that the participants are generally able to determine drunk speakers, we consider this fact less important.

4.5 Conclusion on User State Recognition

Identifying the state of the user for rendering the dialogue management user-adaptive is a rather difficult task. To solve it, we have proposed several novel approaches for automatically recognising different user states and have provide proof of their potential by their successfully application improving the overall recognition performance. The potential, though, highly depends on the user state, e.g., the user satisfaction (on a five-scale) is harder to estimate than the emotion of a person (using only a well-defined subset of emotions).

Furthermore, the user states are estimated on different levels. User satisfaction and the perceived coherence of system dialogue acts are estimated on a turn level using interaction parameters as input variables. Emotions and intoxication, on the other hand, are closer related to speech thus using speech parameters directly for the estimation process. This may be related to the difficulty of the classification task, e.g., links between speech and emotions may be identified much easier compared to user satisfaction and the overall interaction.

For user satisfaction, which we have put an emphasis on in this thesis, we identified two major aspects which are important for the overall performance. Having interaction parameters derived from different modules of the dialogue system, i.e., ASR, NLU, and DM, the DM parameters offering the highest abstraction level contribute most to user satisfaction. This means that, while attributes of the ASR and NLU influence DM, decisions made in the DM have a very high potential to influence the user satisfaction. The second finding is that user satisfaction is not a local phenomenon, i.e., only dependent on the current situation within the dialogue. The situation is quite the opposite: we showed a connection of user satisfaction to the complete interaction up to the beginning which is already reflected in the set of interaction parameters. By exploiting

this relation, we were able to achieve a significant improvement in the recognition performance compared to the state-of-the-art.

While working on user state recognition, it became clear that not all user states are equally applicable for adaptive dialogue management (although all are desirable). While adapting to the user satisfaction is reasonable in a multitude of systems and domains, adapting to the emotions or the intoxication level has more restrictions. Adapting to emotions require a domain (or task), where emotions occur frequently and play an important role in the interaction. For task-oriented dialogues, e.g., getting information about the schedule of the public transit, this might not be the case. This lack is even more evident for intoxication. Hence, for our research on adapting the dialogue to the user state, we focus on the two remaining user states *user satisfaction* and *perceived coherence*. Approaches on how to include these into the dialogue manager will be described in the following chapter.

5

User-Adaptive Dialogue Management

With the ultimate vision of rendering human-machine communication more natural, rendering the dialogue interaction user-adaptive by taking into account the information about the user is one the main goals of this thesis. In the previous chapter, we have focused on the other main goal of how to derive information about the user—the user state—automatically by presenting our work on recognising four user states automatically. Within this chapter, we will propose novel concepts on how to use this information to automatically adapt the course of the dialogue to the user state by rendering the dialogue manager user-adaptive.

In general, two dialogue management adaptation types exist (Nothdurft et al., 2016): adaptation to statically and dynamically changing information. For the latter, the system's behaviour is usually statically influenced, e.g., by user preferences stored in a user model. However, the user state falls into the category of dynamically changing information where the course of the ongoing dialogue is influenced dynamically by the user state. Moreover, adapting the course of the dialogue to the user offers two different types of adaptation:

Adaptation to the user state with short-term goal
: This means adapting the dialogue to the user state derived from the ongoing dialogue and modifying the course of the dialogue to improve the current interaction's general performance, e.g., success, or the user state, e.g., user satisfaction.

Adaptation to the user state with long-term goal
: This means adapting the dialogue to the user state derived from the ongoing dialogue and modifying the ongoing dialogue to reach a long-term goal, e.g., establishing human-computer trust.

Both types have in common that the user state is derived from the interaction and used to influence the action selection. The difference is—for the first—the interaction is adapted to reflect an increase in the target user state directly within the same interaction. For instance, if the user is not satisfied with the interaction, the goal is to increase the satisfaction of the user for the same interaction. This is in contrast to a long-term goal. For example, if the long-term goal is to maintain human-computer trust, it is not necessarily important that the current interaction is going well but that the users feel they can trust the system nonetheless.

For adapting the course of the dialogue to the user state with short-term goal, several modes exists. Ultimately, all lead to using the user state to influence the selection of the next system action. Here, one mode of adaptation is to limit the the pool of system actions to the set of a non-adaptive version of the SDS (Ultes et al., 2011a, 2014a). The user state may then be used

to influence the dialogue strategy, e.g., to answer the question of when to apply explicit confirmation prompts. Of course, all different kinds of dialogue strategy aspects are possible, e.g., the grounding strategy, the dialogue initiative, or the prompt design (cf. 2.1.3).

Another mode of adaptation is to use the user state to trigger special sub-dialogues which then deal with a certain aspect of the user state. As an example, in an emotion-aware dialogue where the user has found to be sad, the system may trigger a sub-dialogue which aims at cheering the user up. Here, extra system actions need to be integrated compared to a non-adaptive version. Other examples are help actions or error recovery strategies triggered by the user state. Within this chapter, though, the focus lies on the former mode as its application possibilities are more general. This is because there is less modelling needed compared to introducing sub-dialogues as those need to be newly defined. For solely influencing the action selection based on an existing pool of system actions, no additional modelling needs to be done.

Adapting the course of the ongoing dialogue to a dynamically derived user state (with a short-term goal) entails certain requirements the user states should match as described in Chapter 4. As all user states which we have considered for our experiments in Chapter 4 comply with the identified requirements, all may be used for this kind of online adaptation.

Among these user states we used for our experiments on user state recognition are *perceived coherence of system dialogue acts* and *user satisfaction*. For the latter, if the system detects that the user is not sufficiently satisfied with the interaction, it may take measures to increase the user's satisfaction level. As user satisfaction and perceived coherence are domain-independent entities which may occur in almost all types of dialogue, this section focuses on adaptation to both. Here, we will propose three novel approaches which will be presented in the first section of this chapter (Sec. 5.1). Following that, we will describe how we have implemented the proposed dialogue management approaches. For this, we had to partially re-engineer an existing dialogue manager which is described in Section 5.2. Afterwards, our novel approaches will be evaluated with real user interaction as well as in a simulated environment for user satisfaction in Section 5.3 and perceived coherence in Section 5.4.

5.1 Approaches for User-Adaptive Dialogue Management

For rendering a dialogue system to be user-centred by taking into account the user state, potentially all possible approaches may be subsumed into a common generic concept which may be visualised with the general processing sequence of a spoken dialogue system. To that effect, the sequence needs to be extended in order to realise any type of adaptation to dynamically changing information and may be viewed as a cyclic process involving the human as one part. For the extension of this dialogue cycle to allow for user adaptation, a new module is introduced (see Figure 5.1) which provides the current user state. Without loss of generality, the cycle may be regarded to start with the system selecting the first system action. This can be seen as valid for all situations if the set of system actions also includes the action of only waiting for user input without producing any output. In general, based on the selected system action, output is created. Based on this system action, the user response is fed as input into the system. Usually, this involves automatic speech recognition and a semantic interpretation module (cf. Sec. 2.1)

For enabling the dialogue system to react adaptively, the user state is required. As thoroughly discussed in Chapter 4, deriving the user state is done without human intervention and automatic

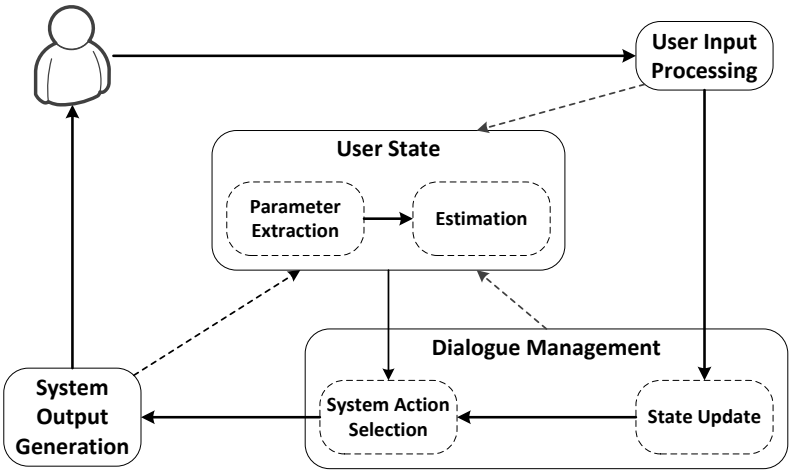


Fig. 5.1: The adaptive dialogue processing cycle originally published in (Nothdurft et al., 2016): for adapting to additional user states, the modules Parameter Extraction and Estimation are integrated producing the estimation of the user state.

estimation approach like statistical classification are used. For this, parameters are needed as input to the estimator and are derived from the interaction taking into account information from all dialogue system components, i.e., speech recognition, semantic interpretation, dialogue management, and output generation. Based on these input parameters, the user state may then be determined and fed into the action selection module of the dialogue management. This user state along with the updated system state are then used to select the next system action and the cycle starts anew.

Having this type of adaptation to dynamically changing information also encompasses several issues. Adding the user state to the action selection may be regarded as an increase in dimensionality and complexity of the problem. This also results in an added uncertainty to the system which should be handled adequately, e.g., using Partially Observable Markov Decision Processes (POMDPs). Furthermore, having several adaptation modes, the question arises which mode is suitable, taking into account the type of adaptation to dynamically changing information as well as the dialogue situation. Finally, our dialogue management concepts which we will present in this section are based on a system where the only modifications of the system state are due to user input.

In the following, we will propose three different methods on how to integrate the user state into the dialogue management process. The first method is based on rule-based policies which entails an extension of the dialogue state. The second method uses the user state within a reward function for reinforcement learning. Finally, we will propose a two-stage approach where the user state is used to select the next system action by predicting the effect of that system action on the user state itself.

5.1.1 User-Adaptive Dialogue Management with Rule-based Policies

Incorporating the user state into the action selection mechanisms of the dialogue manager may be done in several ways—statistical and non-statistical. All have in common that at certain points

within the interaction, the user state may branch the dialogue flow. This branching may be either learned automatically (which will be discussed in the next section) or explicit rules may be defined. These rules then take into account the user state for selecting the next system action. For example, if the user is angry a different user action may be selected then when the user is sad.

Here, of course, different options exist at which point in the dialogue flow the user state may be taken into consideration. While, in general, this may be done at arbitrary points, reasonably, a specific aspect of the dialogue strategy is targeted for adaptation, e.g., the common aspects *dialogue initiative* or *grounding strategy* as presented in Section 2.1.3.

In Figure 5.2, an example is shown for how to adapt the dialogue initiative based on some user state. Here, three different options for the initiative are possible. For a high user state value—whatever that may be or represent, e.g., the user's *happy* state—a system action representing user initiative may be selected. For low user state values, the system will decide to take over the initiative for the following system action. And for all other user state values in between, a mixed-initiative system action is selected.

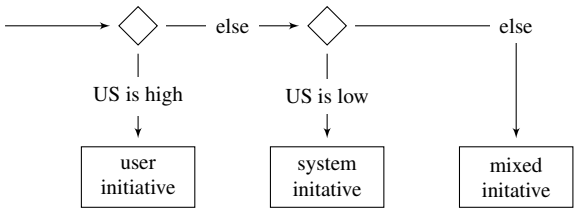


Fig. 5.2: Adaptive rule-based dialogue initiative strategy: based on the user state (US), the type of dialogue initiative for the next system action is selected.

This is similar to adapting the grounding strategy or, more precisely, the type of confirmation prompt as presented in Figure 5.3. Here the type of confirmation prompt is selected based on whether the user state is high, low, or something in between.

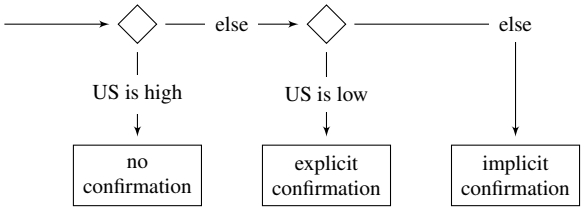


Fig. 5.3: Adaptive rule-based grounding strategy: based on the user state (US), the type of confirmation prompts is selected.

For adapting the strategy to the user state with an information state based dialogue manager (as presented in Section 2.1.2), including the user state and creating user-dependent rules for action selection is straight forward. The user state is simply added to the information state itself. The rules are then realised by adding conditions to the system actions to be executed: only if a certain condition regarding the user state is fulfilled, the corresponding user action may be executed.

Adding information about the user state to the HIS framework is more complex, though, as it results in the need of altering the POMDP formalism. To realise this for SDS-POMDPs as described in Section 2.1.2, the system state $s = (u, g, h)$ is extended by M user states s_m, resulting in $s = (u, g, h, s_1, \ldots, s_M)$. Following the concept of user acts, we further introduce the concept of user state acts e_m, which come along with each user act. This leads to the system state $s = (u, g, h, e_1, \ldots, e_M, s_1, \ldots, s_M)$.

Having the new extended system state incorporating the user state also entails an alteration of the belief update. Building upon the formalism described in Equation 2.5, user states and user state acts are incorporated. Here, act e_m is associated with state s_m. Furthermore, reasonable assumptions about independence of variables are made in accordance to previous work (Williams et al., 2005) leading to

$$\begin{aligned} & b'(u', g', h', e'_1, \ldots, e'_M, s'_1, \ldots, s'_M) \\ &= k \cdot P(o'|u') \cdot \prod_{m=1}^{M} P(o'|e'_m) \cdot P(u'|g', a) \cdot \prod_{m=1}^{M} P(e'_m|s'_m, a) \\ &\quad \cdot \sum_{s_1} P(s'_1|s_1, a) \cdot \ldots \cdot \sum_{s_M} P(s'_M|s_M, a) \cdot \sum_{h} P(h'|u', g', e'_1, \ldots, e'_M, s'_1, \ldots, s'_M, h, a) \\ &\quad \cdot \sum_{g} P(g'|g, a) \cdot \sum_{u} \sum_{m=1}^{M} \sum_{e_m} b(u, g, h, e_1, \ldots, e_M, s_1, \ldots, s_M) \,, \end{aligned} \tag{5.1}$$

where k is a normalising constant.

While the formalism above describes how to introduce user states in general, having a dynamic user state yields to further simplifications: here, the user state and the user state act may be regarded as relating to the same concept as, usually, the user state is estimated directly as described in Chapter 4. Thus, by having $s_m = e_m$, the system state is $s = (u, g, h, s_1, \ldots, s_M)$ and the belief update equation is reduced to

$$\begin{aligned} b'(u', g', h', s'_1, \ldots, s'_M) = \; & k \cdot P(o'|u') \cdot \prod_{m=1}^{M} P(o'|s'_m) \cdot P(u'|g', a) \\ & \cdot \sum_{s_1} P(s'_1|s_1, a) \cdot \ldots \cdot \sum_{s_M} P(s'_M|s_M, a) \cdot \sum_{h} P(h'|u', g', s'_1, \ldots, s'_M, h, a) \\ & \cdot \sum_{g} P(g'|g, a) \cdot \sum_{u} \sum_{m=1}^{M} b(u, g, h, s_1, \ldots, s_M) \,. \end{aligned} \tag{5.2}$$

In accordance to the general notion of POMDPs for SDS, the observation probabilities can easily be estimated using n-best-list input $[(u^1, c_{u^1}), (u^2, c_{u^2}), \ldots]$ where a confidence score c_{u^i} is assigned to each observed user act u_i. This concept may easily transferred to observations and user states: $[(s^1_m, c_{s^1_m}), (s^2_m, c_{s^2_m}), \ldots]$ represents the n-best-list where each user state s^i_m has a confidence value $c_{s^i_m}$. The observation probabilities may then be estimated by $\tilde{P}(o'|u' = u^i) = c_{u^i}$ and $\tilde{P}(o'|s'_m = s^i_m) = c_{s^i_m}$, respectively.

As an example, we take a scenario where the speaker state is 1-dimensional consisting of $s_1 = emo$ representing emotion with the values *neutral*, *angry*, *sad*, and *happy*.

The observation probability $P(o'|emo')$ for the user state *emo* is then represented by the n-best-list of emotion state observations, e.g., the 2-best-list

$$o_{emoAct} = [(angry, c_{angry}), (neutral, c_{neutral})] \,. \tag{5.3}$$

Hence, the estimate of $P(o'|emo')$ is

$$\tilde{P}(o'|emo' = angry) = c_{angry} \tag{5.4}$$
$$\tilde{P}(o'|emo' = neutral) = c_{neutral} \tag{5.5}$$
$$\tilde{P}(o'|emo' = remainder) = c_{remainder}. \tag{5.6}$$

Here, *remainder* combines all remaining emotions and has the remaining probability mass not contained in c_{angry} and $c_{neutral}$: $c_{remainder} = 1 - c_{angry} - c_{neutral}$.

Having only one user state *emo*, belief update is reduced to

$$\begin{aligned} b'(u', g', h', emo') = \; & k \cdot P(o'|u') \cdot P(o'|emo') \cdot P(u'|g', a) \\ & \cdot \sum_{emo} P(emo'|emo, a) \cdot \sum_{h} P(h'|u', g', emo', h, a) \\ & \cdot \sum_{g} P(g'|g, a) \cdot \sum_{u} b(u, g, h, emo) \,. \end{aligned} \tag{5.7}$$

Until now, we have shown how states may be included into the general formalism for SDS-POMDPs. However, the problem still exists that the belief update is computational intractable, especially if extending the system state with user states. Here, further simplifications based on assumptions of independence are introduced: we assume that the user state s_m is independent of the user act u, the user goal g, the history h, and all other user states s_j where $j \neq m$. While the degree of independence varies depending on the user state, e.g., the emotion may be less influenced by u, g, and h than the user satisfaction, the correlation may be regarded as not high or being reflected somewhere else, e.g., within the interaction parameters used for recognising the user state. This simplification finally leads to two equations for tracking the two belief states $b(u, g, h)$ and $b(s_m)$ where the former is identical to Equation 2.5. The latter is defined as

$$b'(s'_m) = k \cdot P(o'|s'_m) \cdot \sum_{s_m} P(s'_m|s_m, a) \cdot b(s_m) \tag{5.8}$$

for $m \in \{1, \ldots, M\}$. This equation is then identical to Equation 4.18 where the user satisfaction is estimated using a Hybrid-HMM approach also taking into account the last system action.

By introducing this simplification, known approaches for rendering belief tracking tractable may be applied without any problems. More precisely, we are using the HIS approach (see Sec. 2.1.2) within this thesis which renders the belief update tractable by partitioning the user goal space into equivalence classes or partitions. Each partition then has a probability associated indicating their belief value. To apply rule-based adaptation within the HIS framework, simply the partition with the highest probability is selected. Then, based on the partition's content, the rules may be applied.

Of course, besides applying rule-based adaptation techniques which are based on an extended dialogue state, another major part of formulating the dialogue management problem as a POMDP is to apply automatic approaches for learning an optimised policy. Hence, in the following section, we propose our ideas of how the user state may be used within those methods.

5.1.2 User-Adaptive Dialogue Management with Policy Optimisation

In contrast to our proposed approaches described in the previous section where rules have been defined manually to adapt the dialogue to the user state, we focus in this section on methods which automatically learn optimal strategies based on the user state. As already outlined in Section 2.1.2, reinforcement learning techniques are used to automatically learn an optimal policy $\pi^*(b)$ for each belief state b. For POMDPs, learning the optimal policy for each belief state, though, is a very complex problem. Williams and Young (2005) proposed to map the belief state b on a summary belief state $\hat{b}$. For the HIS methodology, the summary belief state is extracted out of the partition distribution (Young et al., 2010). More precisely, it consists of

- the belief value of the top partition,
- the belief value of the second-top partition,
- the last user action,
- the partition state, and
- the grounding state.

Partition and grounding state indicate the general status, e.g., if the partition is consistent with a single unique matching entity or entities of the partition have been confirmed by the user.

Furthermore, Young et al. (2010) propose a grid-based policy optimisation algorithm. Here, the policy is stored as a set of grid points within the summary belief space. During execution time, for each belief state b, the next system action a is identified by mapping finding the closest grid point and executing the associated action. However, as the grid points are in the summary space, only summary actions $\hat{a}$ are selected, e.g., `request` without slot information. These summary actions are then transferred to a regular system action using a heuristic. Based on this, the Algorithm 1 outlined in Section 2.1.2 may be used to learn an optimal policy.

Finding an optimal strategy using a grid-based approach, though, results in the need of huge amounts of training dialogues (> 100.000). Hence, Gačić et al. (2010) proposed Gaussian process learning. In contrast to the grid-based approach, the Q-function (Eq. 2.10) modelling the expected cumulative reward for the current state-action pair is approximated by Gaussian distributions. Hence, by sampling the distribution given the current summary state, the next summary action may be derived. This has been described in more detail in Section 2.1.2.

For incorporating the user state into both learning algorithms, there exist two principle options. The first option is—as described in the previous section—to add the user state to the system state. To be reflected in the learning algorithm, it also needs to be part of the summary belief space. Then, the learning algorithm will automatically take the user state into account when comparing summary belief points and learn an optimal strategy accordingly.

The second option is to use the user state for reward modelling. For most state-of-the-art POMDP-based dialogue systems, the reward function is defined to be rather simple: for each system action, the reward is -1. This is to enforce short dialogues. In the end, a high positive or negative reward is added depending on whether the dialogue was successful, e.g., +20 for success or -50 for failure.

Using the user state offers further interesting options. For example, it may be used to define whether the final reward is applied, i.e., whether the dialogue was successful or not thus offering an alternate notion of success. While this decision may usually only automatically be taken when learning with simulated dialogues, using an automatically derivable user state offers the same functionality for interaction with real users. In the case of user satisfaction, e.g., the dialogue may

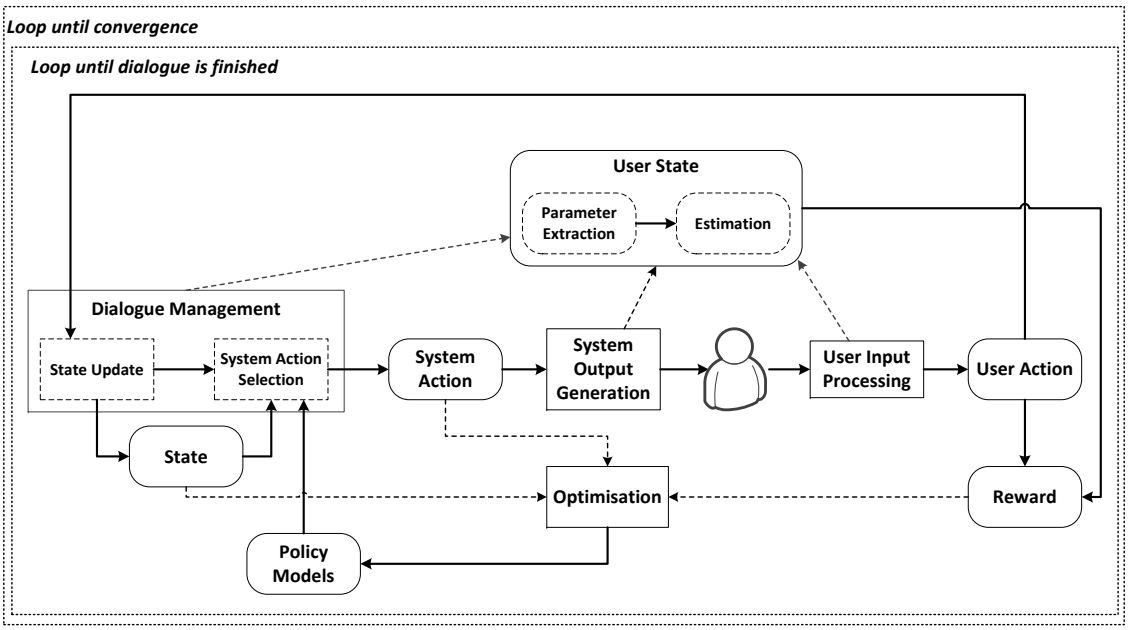


Fig. 5.4: The adaptive dialogue cycle for learning an optimal strategy using GP-SARSA algorithm described in Algorithm 2. At each turn the system action, the updated dialogue state distribution and the reward are used by the optimiser (along with the system action and the dialogue state of the previous turn) to update the policy model.

be regarded as successful if the user is satisfied up to a certain degree in the end, as unsuccessful otherwise.

Another option for using the user state for reward modelling is to take it into account directly, e.g., for rewarding each system action (thus replacing the generic -1 value). In the case of user satisfaction, which is modelled on an integer scale from one to five, the value may be used directly, e.g., $r = -6 + s_{us}$. Here, exchanges where the user is not completely satisfied ($s_{iq} < 5$) are discounted. Furthermore, changes in the user state may also be used, e.g., if the user's emotion state changes from *sad* to *neutral*, this may be rewarded with a positive value. Naturally, the reward highly depends on the general task of the dialogue system.

To include both types of reward modelling into the GP-SARSA algorithm, the adaptive dialogue cycle is altered for the learning phase. The overall scheme is depicted in Figure 5.4. In the centre, the known cycle is depicted with added system action, user action, state, and reward. Here, the state, the system action, and the reward are then used (along with the system action and the dialogue state of the previous turn which have to be stored) directly to optimise the policy. This updated policy may then be used for selecting the next system action directly. Furthermore, the user state is identified for each system-user exchange and fed into the reward.

For both types of user state adaptation—extending the dialogue state and learning an optimised policy—we will provide evidence for their potential in user-adaptive dialogue modelling in Section 5.3. A third way of using the user state has also been explored: predicting the influence of executing a candidate system action on the user state. This will be described in the next section.

5.1.3 Re-ranking of System Actions Based on the User State

With the general goal of rendering the dialogue manager user-centred, we have already proposed two approaches on incorporating the user state: by extending the dialogue state and by learning a user state-based optimal policy. However, the user state may also be used to influence the selection of the next system action, or system dialogue act, by predicting the influence of said system action on the user state.

For the user state of perceived coherence, we propose a two-stage approach shown in Figure 5.5: first the dialogue manager produces an n-best list containing possibly suitable dialogue acts for the next system response. Then, the final dialogue act is selected according to an estimation of the coherence of all n dialogue acts taking into account the current dialogue situation. We propose to use the procedure in Figure 3 to compute the final DA: if the best dialogue act chosen by the dialogue manager is coherent, then it is selected as the next system response. Else, the n-best list is re-ranked according to their estimated coherence. Finally, the first-best entry of the re-ranked list is chosen.

This approach aims at addressing the problem of generating coherent system responses in human-machine-communication, i.e., the system response should be coherent with the conversation up to the current moment. Here, utilising the coherence information extracted from the conversation may help. By rendering the approach as a two-stage process thus adding a successive component after the dialogue manager, virtually any approach for dialogue management may be applied without limiting the applicability of our proposed method.

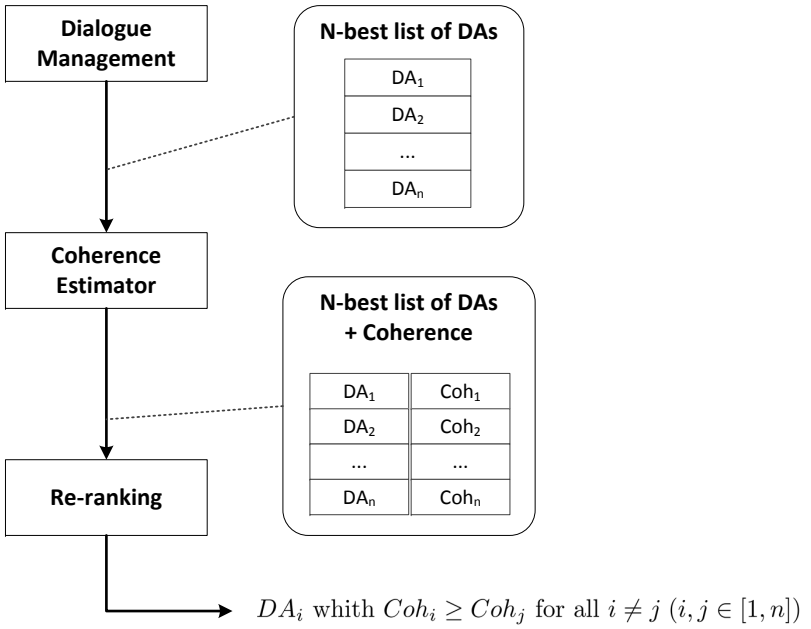


Fig. 5.5: Schematic overview of the two-stage coherence-based dialogue management approach

Hence, this procedure is portable to different application domains and may involve a diversity of dialogue managers and automatic coherence estimation approaches. Therefore, it can be easily adopted in existing systems with the only requisite that the dialogue manager generates an n-best

Algorithm 3: Enhanced dialogue management procedure

```
1 begin
2     DAs ← statical_dialogue_management() ;
3     DAs_annotated_coherence ← coherence_estimator(DAs) ;
4     if DAs_annotated_coherence[0] = "Stronglycoherent" then
5         return DAs[0] ;
6     else
7         return select_most_coherent(DAs_annotated_coherence) ;
```

list of system DAs and the coherence estimator receives this lists and assigns a coherence label to each alternative.

Before we evaluate our proposed approaches for user-adaptive dialogue modelling, we will continue with describing the implementation of the proposed methods by extending the used dialogue manager with the hidden information state and the user state recognition module.

5.2 Implementation of the User-Adaptive HIS-OwlSpeak Dialogue Manager

While we have proposed three genuinely different approaches of how to include the user state into the dialogue manager in the previous sections, in this section, we will continue with a thorough description of how the user state recognition module—necessary for including the user state information into the dialogue manager—as well as the hidden information state POMDP approach are implemented. As basis for this implementation, we will use the dialogue manager OwlSpeak presented in Section 2.1.2. To integrate our proposed methods, we will partially re-engineer and re-design core components of OwlSpeak. The resulting dialogue manager will then offer different control modes (thus being able to adapt internally to domains of different complexity, (cf. Ultes and Minker, 2014b)). Furthermore, it will offer different internal system state representations by implementing the HIS approach into OwlSpeak. As the basic OwlSpeak system is based on the IS approach, applying the hidden information state approach—an extension of the IS—for introducing probabilistic dialogue into OwlSpeak seems natural. The resulting system will have the capability of providing different operation modes concerning the number of state hypotheses and the policy:

- Rule-Based Control + Single State Hypothesis
- Rule-Based Control + Multiple State Hypotheses
- Trained Policy-Control + One State Hypothesis
- Trained Policy-Control + Multiple State Hypotheses

In accordance to the presentation of the original OwlSpeak in Section 2.1.2, the re-design necessary for our implementation of a user-adaptive dialogue manager is described by following the Model-View-Presenter paradigm. Subsequently, we will describe how the user extraction module was integrated into the dialogue manager.

5.2.1 Spoken Dialogue Ontology (Model)

Integrating our proposed dialogue management approaches into OwlSpeak to include HIS functionality, the model has to be partially re-designed by adding or altering ontology concepts leading to a modified ontology description. The new ontology is shown in Figure 5.6 with all new concepts and relations coloured dark grey. Analysing the spoken dialogue ontology (SDO) of OwlSpeak reveals that most needed concepts already exist. This is not surprising as the HIS approach is an extension of the IS approach. The *BeliefSpace* concept represents the same functionality as a partition does. Only some relations are added: *parent* and *children* relations are necessary in order to create the hierarchical structure of the HIS partition space. Furthermore, assuming a slot-filling dialogue, the relation *excludesBelief* represents all slot values which are excluded in this partition. A slot taking one specific value, on the other hand, is represented by the *hasBelief* relation.

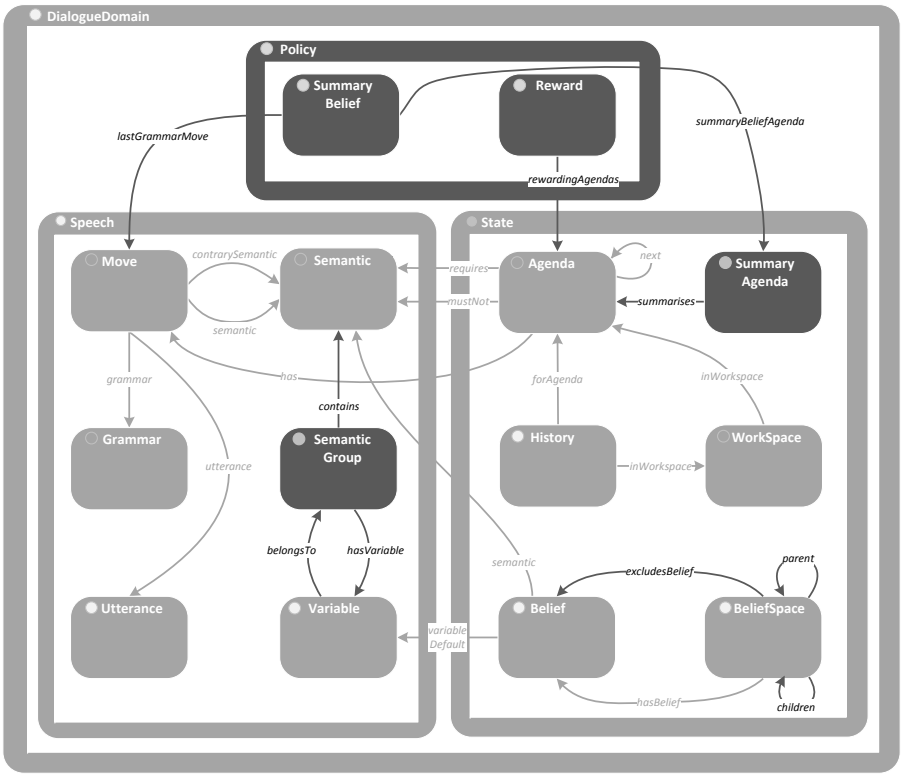


Fig. 5.6: The scheme of the Extended Spoken Dialogue Ontology (SDO) originally published in (Ultes et al., 2014a). Concepts belonging to the original SDO are light grey while concepts and relations introduced for the HIS implementation are dark grey. The static dialogue description is shown on the left side of the picture within the *Speech* class while the concepts belonging to the dynamic *State* of the system are shown on the right side. Additionally, concepts belonging to the *Policy* are shown at the top.

The basic version of OwlSpeak does not support slots directly. For realising a slot-filling dialogue in the basic implementation of OwlSepak as described in Section 2.1.2, the expert designing

the dialogue needs to handle this, e.g., by using naming conventions. However, no relations between semantics or variables may be modelled explicitly within the ontology. For a dialogue in the flight booking domain, for example, a semantic representing a concrete slot value for the slot "destination" would be `sem_destination_miami`, representing the semantic expressing that the desired destination is Miami. The slot is "destination" and only encoded in a textual manner. Of course, a general semantic for the destination may be added as well. However, there is no way of modelling any connection explicitly.

The HIS theory is based on a slot concept, though. Hence, the ontology class *SemanticGroup* is introduced into the *Speech* class representing one slot. A semantic group may contain multiple semantic objects. Furthermore, a semantic group may also be related to a *Variable* by the *hasVariable* and *belongsTo* relations. In order to reflect the slot mechanism using semantic groups in the belief space, the semantic group may also be contained as a *Belief* individual in the belief space. To accomplish this, each time a new *Belief* individual is written to the belief space containing slot value information, the corresponding semantic group individual is also written to the belief space if it does not already exist there.

With the presented structure for slot functionality in OwlSpeak, two modelling variants are realised. First, each slot value may be represented with its own semantic individual all connected to the same semantic group. By modelling these slot values explicitly, they all may easily be collected out of the ontology. Furthermore, each semantic may contain further information, e.g., how to name this value in speech synthesis for multiple languages. The drawback of using the *Semantic* concept is that, for slots with many values, many semantic individuals will be created increasing the size of the ontology accordingly. The second variant accounts for that by utilising the *Variable* concept to represent slot values. The value is then written into the variable reducing the number of ontology individuals to one. However, no additional information for the respective slot value may be stored. Thus, multi-linguality for speech synthesis as mentioned above, for instance, is much harder to achieve.

Furthermore, the concept *SummaryAgenda* is added to the *State* class of the SDO. It groups all agendas with similar type, e.g., "requesting information" or "asking for confirmation". The *SummaryAgenda* is related to a set of agendas, implemented by the *summarises* relation.

Finally, a new general class is added to the *DialogueDomian* called *Policy*. It is necessary for enabling OwlSpeak to create optimised policies as outlined in Section 2.1.2 and described by Young et al. (2010). The *SummaryBelief* class contains fields representing the belief values of the two top partitions as well as the last grammar move, i.e., the last user action. Furthermore, by mapping a computed temporary summary belief—representing a summary of the current partition distribution of the running system—to a *SummaryBelief* belonging to the policy, a summary system action is selected which is determined by the *summaryAgenda* relation of the policy summary belief individual. Out of the related set of agendas, a suitable agenda is selected by applying a heuristic. This is elaborated in Section 2.1.2. In order to enable the usage of methods for creating optimised policies, a reward function needs to be defined. This is realised by the *Reward* class defining a reward value for all agendas which are connected by the *rewardingAgendas*.

5.2.2 Voice Document (View)

While the implementation of OwlSpeak described in Section 2.1.2 is based on single user input (even without regarding the confidence score of the ASR interpretation), implementing the HIS

entails n-best list functionality for the user input. As VXML documents are used as interface between the dialogue manager and the remaining dialogue system components, the created documents need to be altered to add n-best list functionality. Fortunately, VXML provides mechanisms for using n-best list with confidences inherently. N-best-lists are activated by adding

```
<property name=''maxnbest'' value=n/>
```

to the element `<form>`. To access this new information, the VXML shadow variables

```
resultArray[i].interpretation
resultArray[i].confidence
```

are used. `resultArray[i]` refers to the `i`-th element of the n-best list of ASR interpretation results. The collected n-best list is then added to the field list of the `<submit>` element.

As we will also conduct experiments where a user simulator (see Sec. 2.1.4) is used instead of real users, the view is altered in order to allow communication with the user simulator.

5.2.3 Dialogue Control (Presenter)

Adding to OwlSpeak the ability of offering different control modes and different types of system states entails serious modifications to the control system. While the original algorithm of OwlSpeak may be used having only a single state hypothesis, handling multiple state hypotheses is very complex. Therefore, the AT&T Statistical Dialogue Toolkit (ASDT) (Williams, 2010a) is used.

The general idea of updating the system state with a single hypothesis has already been described in Section 2.1.2. A formalisation of updating the state is shown in Algorithm 4. As only one user input may be processed, first, the entry of the n-best list with the highest confidence is selected. This user action is represented by a *Move* individual hence including *semantic* and *contrarySemantic* relations as well as variable operators. All of these are handled: First, beliefs containing semantics of the *contrarySemantic* relation are removed from the belief space, second, new beliefs for semantics of the *semantic* relation are added, and finally the variables are handled by creating belief individuals in the belief space and adding the variable value as well as the *variableDefault*. Of course, corresponding *SemanticGroup* individuals are also added or removed wherever applicable.

Algorithm 4: State update (single hyp.) as published in (Ultes and Minker, 2014b)

Data: Belief space *bs*, list of user actions $\tilde{\mathbf{u}}$, their confidences $c(u)$
Result: Updated system state (belief space *bs*)

```
1 begin
2     u ← arg max_{u∈ũ} c(u)
3     Remove contrarySemantics of u from bs Write semantics of u to bs
4     Set variables of u in bs
```

As already mentioned above, for handling multiple state hypotheses by using the Hidden Information State approach, the ASDT is used. It contains all HIS concepts presented in Sec-

tion 2.1.2 and further conveys additional efficiency for tracking multiple dialogue states by incremental partition recombination as presented by Williams (2010b).

The general mechanism for updating the partition distribution is already provided by the ASDT (see Algorithm 5). For each user action of the n-best list and each partition, the partition is split using Algorithm 6 followed by an update of the belief value, i.e., probability, of the current partition. If the total number of partitions exceeds a predefined maximum number, all leaf partitions with minimum probability are merged with their parents until the total number of partitions is less than the predefined value.

Algorithm 5: State update (multiple hyp.) as published by Williams (2010b)

Data: Set of partitions $\mathbf{p}$, their beliefs $b(p)$, list of user actions $\tilde{\mathbf{u}}$, parameter p_{max}
Result: Updated set of partitions $\mathbf{p}$ and beliefs $b(p)$

```
1 begin
2     for n ← 1 to |ũ| do
3         for m ← 1 to |p| do
4             If possible, split p_m on u_m and add children to p
5             Calculate new belief b'(p_m)
6         while |p| > p_max do
7             p ← arg min_{p:p is leaf} b'(p)
8             Recombine p with parent and remove from p
9     Normalise b'(p)
```

Partition splitting is rendered by Algorithm 6. It decides whether the partition is split, i.e., children are created, or the current partition is solely updated. The latter happens if the user action does not represent new slot information but only a confirmation move. Then, if the user action matches the information of this partition, the confirmation information is inserted into the partition (represented as a *Semantic*). A partition matches a user action if they do not disagree, e.g., if the slot value of the user action is not excluded by the partition.

A new child partition is created using Algorithm 7 if the user action does not represent a confirmation move but contains new slot information. By definition, the slot value of the user move is added to the excludes list of the parent partition while the slot in the newly created child partition equals the value (cf. Section 2.1.2). Hence, a new belief is created in the child partition and added to the *hasBelief* relation. All beliefs which do not relate to the given slot are left untouched and simply copied to the child partition.

Furthermore, calculating the new belief value (Line 5 of Algorithm 5) is also mainly handled by the ASDT. Only the probability $p(u'|p',a)$ of the user action given the partition and the system action (cf. Equation 2.6) needs to be computed. Ideally, a real probability distribution should be learned. However, implementations in real systems have shown that this is not necessary. It usually suffices to use a general algorithm differentiating the important cases. The resulting Algorithm 8 hence returns a probability of 0.0 if the user action disagrees with the partition and a 1.0 if the user action agrees. As a user action may address more than one slot, the likelihood for each slot is determined separately and the final likelihood is normalised over all slots.

Algorithm 6: Split partition as published in (Ultes and Minker, 2014b)

```
Data: User action u, current partition p
Result: Updated p, child partition p_c
1 begin
2     if u is confirmation then
3         if u matches p then
              // no split
4             Update p by inserting confirmation
5     else
6         if u matches p then
              // split
7             Create new child partition p_c
```

Algorithm 7: Create child as published in (Ultes and Minker, 2014b)

```
Data: Parent partition p, slot s with value v
Result: Newly created partition p' and updated parent p
1 begin
2     p' ← empty partition
3     Add v to exclude list of p for slot s
4     Add v of s to p'
5     B ← beliefs in p
6     foreach b ∈ B do
7         if b does not belong to s then
8             Copy b to p'
```

Once the system state is updated, selecting the next system action, i.e., agenda, may be performed. This selection process may be regarded as a function called policy defined by $\pi(s) = a$, where s is the current system state and a the resulting system action. The system offers two policy variants. First, a rule-based policy and second an automatically trained and optimised policy determined in a training phase with hundreds of thousands of example dialogues and associated reward function.

The rule-based policy, as described in Algorithm 9, is identical to the policy created for the original OwlSpeak dialogue manager. In case of having a partition distribution, the partition with the highest belief is selected. For this partition or—in case of only having one state hypothesis—the *BeliefSpace* individual, the set of active agendas from the workspace is checked. As described in Section 2.1.2, each agenda has execution preconditions, e.g., semantics which must or must not be present in the current system state. Out of the resulting set of agendas, the agenda with the highest priority is selected.

As previously described, there are many different approaches known for creating an optimised policy. However, the optimisation problem is complex and standard algorithms often lead to intractable solutions domain which are more complex, e.g., than having two slots and three values.

Algorithm 8: User action likelihood as published in (Ultes and Minker, 2014b)

Data: User action u, its slots $\mathbf{s_u}$, current partition p, its slots $\mathbf{s_p}$, system action a
Result: Likelihood lh of u and a for partition p

```
1  begin
2      foreach s ∈ s_u do
3          if s ≠ ∅ then
4              v ← valueOf(s)
5              s_p ← corresponding slot in u_p
6              if a is confirmation request then
7                  cf ← confirmationInfo of s
8                  if cf is confirm then
9                      if v ∈ s_p then
10                         lh ← lh + 1.0
11                     else
12                         lh ← lh + 0.0
13                 else if cf is reject then
14                     if v ∈ s_p then
15                         lh ← lh + 0.0
16                     else
17                         lh ← lh + 1.0
18             else if a is slot request then
19                 if v ∈ s_p then
20                     lh ← lh + 1.0
21                 else if v excluded by p then
22                     lh ← lh + 0.0
23                 else
24                     lh ← lh + 1.0 / nbRemFieldVals
25     lh ← lh / nbFields
```

Therefore, a grid-based approximation method and a Gaussian process-based method are used. Young et al. (2010) proposed the grid-based method and the according policy implementation is presented in Algorithm 10. It is based on sample summary belief points representing a set of optimal system actions. For selecting the next agenda to be executed, a temporary *SummaryBelief* individual is created. Then, the closest sample from the *SummaryBelief* individuals stored in the ontology is found and its corresponding *SummaryAgenda* is derived. As an heuristic for deriving the *Agenda* individual to be executed, the agenda is selected out of those which are summarised by the *SummaryAgenda* according to their preconditions and priorities. This works similarly to the procedure described in Algorithm 9. However, the semantics of the *mustNot* relation are not regarded. For testing these preconditions, again, the partition with the highest probability is selected. For this implementation, the optimised policy only works with multiple state hypotheses. Having only one system state hypothesis would result in a different configuration of the summary

Algorithm 9: Rule-based policy as published in (Ultes and Minker, 2014b)

Data: Set of partitions **p**, their beliefs $b(p)$, set of agendas available for execution **W**, their priorities $prio(a)$
Result: Agenda $\bar{a} \in \mathbf{w}$ with conditions fulfilled and highest priority

```
1  begin
2      A ← ∅
3      p̄ ← arg max_{p∈p} b(p)
4      foreach a ∈ W do
5          t ← false
6          t ← checkRequiresCond(a,p̄)
7          t ← checkMustNotCond(a,p̄)
8          t ← checkVariableCond(a,p̄)
9          if t then
10             Add a to A
11     ā ← arg max_{a∈A} prio(a)
```

belief points which are not modelled in the current implementation. For learning this optimal policy, Young et al. propose a Monte Carlo ε-greedy algorithm as presented in Section 2.1.2.

Algorithm 10: Grid-based policy as published in (Ultes and Minker, 2014b)

Data: Set of partitions **p**, their beliefs $b(p)$, set of agendas available for execution **W**, their priorities $prio(a)$, set of summary belief points of the trained policy **SB**
Result: Agenda $\bar{a}$ with conditions fulfilled and highest priority

```
1  begin
2      tmpSB ← convertToSummaryBelief(p)
3      closestSB ← findClosestSB(tmpSB,SB)
4      summaryAgenda ← getFrom(closestSB)
5      A ← ∅
6      p̄ ← arg max_{p∈p} b(p)
7      foreach a ∈ summaryAgenda do
8          if a ∈ W then
9              t ← false
10             t ← checkRequiresCond(a,p̄)
11             t ← checkVariableCond(a,p̄)
12             if t then
13                 Add a to A
14     ā ← arg max_{a∈A} prio(a)
```

For the experiments conducted in this work, though, we will rely on a learning method based on Gaussian process function approximation as already described in Section 2.1.2. Here, we will

use the GP-SARSA algorithm as presented by Gačić and Young (2014) for dialogue system policy optimisation. The implemented policy is described in Algorithm 11. Based on a set of summary beliefs, which are stored in the ontology, and two matrices, which are stored separately, the Q-function is approximated. Based on this approximation, the summary agenda having the maximal Q value is selected. It should be noted that, for this selection, only summary agendas are used that are known to contain at least one agenda which requirements are fulfilled. Then, the final agenda is selected randomly out of all agendas contained in the summary agenda with fulfilled requirements.

Algorithm 11: GP-SARSA policy

Data: Set of partitions $\mathbf{p}$, their beliefs $b(p)$, set of agendas with conditions fulfilled $\mathbf{W}$, Gaussian process model for Q-function $Q(c,a)$
Result: Agenda $\bar{a}$ with conditions fulfilled

```
begin
    c ← convertToSummaryBelief(p)
    SW ← extractSummaryAgendas(W)
    summaryAgenda ← arg max_{a∈SW} Q(c,a)
    A ← ∅
    p̄ ← arg max_{p∈p} b(p)
    foreach a ∈ summaryAgenda do
        if a ∈ W then
            t ← false
            t ← checkRequiresCond(a,p̄)
            t ← checkMustNotCond(a,p̄)
            t ← checkVariableCond(a,p̄)
            if t then
                Add a to A
    ā ← getRandomAgenda(A)
```

All control modes, i.e., the rule-based and the optimised policy, are designed to use the same underlying dialogue description thus making it independent of the dialogue domain. Therefore, the same dialogue may be executed using the mode which suits the specific needs of the task best. Furthermore, if multiple tasks are handled by the system, the control mode for each task can be defined separately and independently. Hence, internal adaptivity to the complexity of the task is realised. Moreover, by being able to switch the control mode, the dialogue manager offers an easy way of collecting data using a handcrafted strategy which may be used for policy training afterwards.

While we have explained how the HIS was implemented and integrated into OwlSpeak, still, no user-adaptivity has been achieved as there are no means yet for deriving the user state. Hence, a user state recognition module is added. The implementation of this user state recognition module into OwlSpeak will be described in the following.

5.2.4 Introducing the User State into OwlSpeak

For employing user state adaptivity for dialogue management, which is necessary for changing the course of the dialogue according to the user state, the ontology-based dialogue manager OwlSpeak has been extended. In this section, we will describe this extension exemplary for the user state *user satisfaction* represented by the Interaction Quality (IQ). This process is similar for other user states.

To enable OwlSpeak to being able to adapt the dialogue to IQ dynamically, i.e., during the ongoing interaction, a component deriving the interaction parameters is created. It is designed to store all information which is necessary to derive all interaction parameters on all levels. In order to compute window and dialogue level parameters, a data structure is used internally to store the information for all exchanges up to the current one of the ongoing dialogue. The parameters are computed just in time when they are needed as input to the IQ recognition module.

Comparing the adaptive dialogue cycle (Fig. 5.1) with the architecture of OwlSpeak leads to the conclusion that the interaction parameter extraction module is placed between the "User Input Processing" and "State Update". Hence, it is either located at the end of the view layer or at the beginning of the presenter layer. Out of implementation reasons, the interaction parameter module is placed at the beginning of the presenter layer. There, the interpreted input from the voice browser can immediately be used for updating and calculating the interaction parameters.

However, the information provided by the voice browser is not sufficient for extracting all interaction parameters as some are not necessary for regular dialogue management (e.g., user turn length or ASR confidence). Hence, the view layer is modified by altering the VXML-creation. New variable tags are added and filled with the respective information. Furthermore, the submit tags are altered to provide this information to the dialogue manager once the VXML document is processed completely. This also describes the principle concept of how the VXML document functions as interface between the voice browser and OwlSpeak.

Once the interaction parameters are extracted and calculated, the IQ estimation takes place. The parameter vector is fed into a Support Vector Machine which has been chosen as statistical model as described in Section 4.1.2. The classifier is based on the LibSVM implementation (Chang and Lin, 2011) using a linear kernel. The SVM model is created beforehand which is an advantage as this creation process is time consuming. Hence, only evaluating the input vector consisting of the interaction parameters happens during dialogue run-time.

For creating dialogues which are adaptive to IQ, the IQ value needs to be accessed during creation time. For this, the ontology concept *Variable* is used. The variable "InteractionQuality" is created which may be used within conditions for defining requirements of agendas. These requirements are processed and evaluated during the action selection process and only agendas may be executed whose requirements are fulfilled. Hence, IQ-adaptivity is achieved by adding new preconditions to the system actions. During run-time, the result of IQ estimation is stored within the variable before it is processed in the action selection module.

An example for a conditioned agenda can be seen in Figure 5.7. Having a dialogue in the train booking domain, the agenda combines the concept of implicitly confirming the day of travel and asking for the desired time of departure. It is only executed if the Interaction Quality variable holds a value greater than one. This condition is shown in the "variablesOperator" field using a `REQUIRES` function. Furthermore, other preconditions are visible (in the "mustnot" field) along with a list of moves belonging to this agenda (in the "has" field).

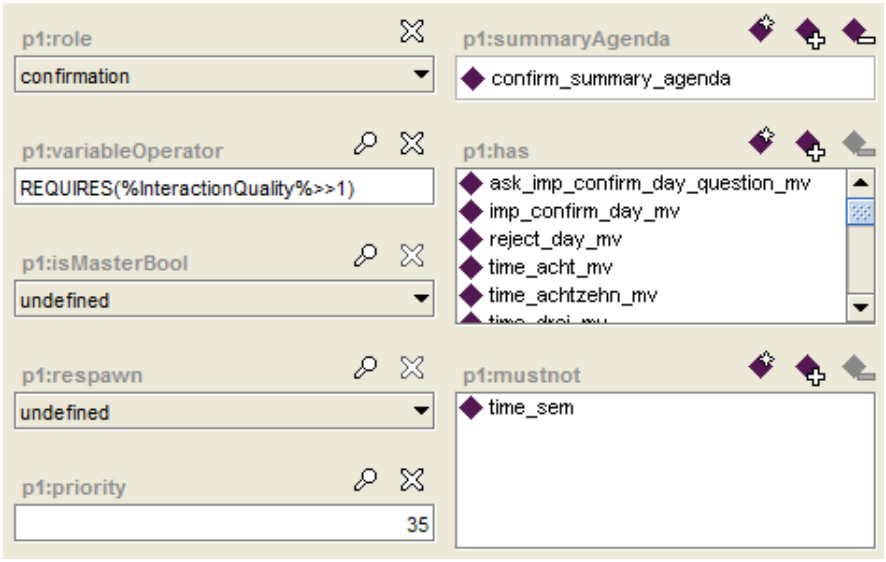


Fig. 5.7: An example of a conditioned agenda originally published in (Ultes et al., 2014a): the variableOperator field contains the requirement that the Interaction Quality must be greater than one for this agenda to be executed.

Now, the implementation and all necessary steps have been described for using the hidden information state together with the user state information for adaptive dialogue management. For the showcase of user satisfaction, we have conducted several studies using the implemented dialogue manager which will be described in the next section. Afterwards, the re-ranking of system actions based on perceived coherence will be evaluated.

5.3 Experiments and Evaluation of User Satisfaction Adaptation

For evaluating the adaptation of the dialogue strategy to the user state, we have proposed several potential user states in Chapter 4. As already stated, the user satisfaction poses the one with the highest possible impact as the users may be satisfied of unsatisfied with the interaction in nearly all human-machine communication settings. Hence, we use the user satisfaction and present experiments in this section on adapting the course of the dialogue accordingly.

To model the dialogue user-adaptive, we have proposed approaches on how to include the user state into the dialogue management process. Here, we have developed three different ideas: applying a rule-based policy (Sec. 5.1.1, learning an optimal policy (Sec. 5.1.2), and re-ranking of system actions (Sec. 5.1.3). In this section, we will present experiments and evaluation for the former two approaches. We will start with a pilot user study on adapting the grounding strategy to user satisfaction followed by an experiment on adapting the initiative. Finally, we will draw some

implications on using user satisfaction for reward modelling in reinforcement learning strategy optimisation.

5.3.1 Pilot User Study Adapting the Grounding Strategy

Introducing user satisfaction adaptivity into the dialogue manager for the first time has unknown potential. While we expect this potential to be very high, still, this should be demonstrated. Hence, we conducted a pilot user experiment introducing a simple adaptive strategy. Based on the extended OwlSpeak dialogue manager described in the previous section, an adaptive dialogue within a simple train booking domain has been created. Depending on the current user satisfaction represented by the Interaction Quality value, the grounding strategy was adapted, i.e., each time the system requests a confirmation about a certain slot value from the user, the user satisfaction value represented by the Interaction Quality (IQ) value is used to decide whether the system uses an explicit or implicit confirmation prompt. In the following, the design and setup of the study will be presented before giving details about the results.

Design and Setup

For adapting the dialogue to the Interaction Quality, the confirmation strategy was selected out of one simple reason: it is an easily adaptable concept which occurs in almost every dialogue in which the user is requested to provide information. A dialogue in the train booking domain was created asking the user for information about the origin, the destination, the day of the week and the time of travel. The user could choose out of 22 cities which were used as origin and destination alike. Furthermore, the time of travel was restricted to every full hour (1 pm, 2 pm, 3 pm, etc.). Three different dialogues were created: one only applying explicit confirmation (all-explicit), one applying only implicit confirmation (all-implicit), and one adapting the confirmation type to the current IQ value (adapted). Besides these differences, the dialogues were the same. The complete dialogue was system initiated and the course of the dialogue was predetermined, i.e., the order of information the user was asked to provide was given. A sample for the adapted strategy is illustrated in Figure 5.8. As only two different options for adapting the dialogue exist, i.e., either selecting implicit or explicit confirmation, the IQ value has been limited to only two values: two representing a satisfied user and one representing an unsatisfied user. If the user was recognised as being satisfied with the dialogue (high IQ value), slot values were confirmed implicitly while explicit confirmation was applied for unsatisfied users (low IQ value). In the end of the dialogue, the user was provided with a dummy message stating that the reservation has been made.

Before the experiment, each participant was presented with a sheet of paper stating all options they could say during the dialogue. This also included a list of all cities. Furthermore, each user participated in three runs of the dialogue—one for each type of confirmation strategy. During the experiment, the order of these dialogues has been alternated to get an equal distribution over all combinations so that learning effects are taken account of. However, the user was not aware about the different dialogue types. After each dialogue, the participants were asked to fill out a questionnaire based (see Fig. A.1) on the SASSI questionnaire (Sec. 2.2.3) to evaluate their overall impression with the dialogue. Each item was rated on a seven-point scale.

In total, there were 24 participants (8 female, 16 male) creating 72 dialogues with an average number of turns of 33.58. They were between 19 and 38 years old with an average age of 26.42. The participants were students from multiple disciplines.

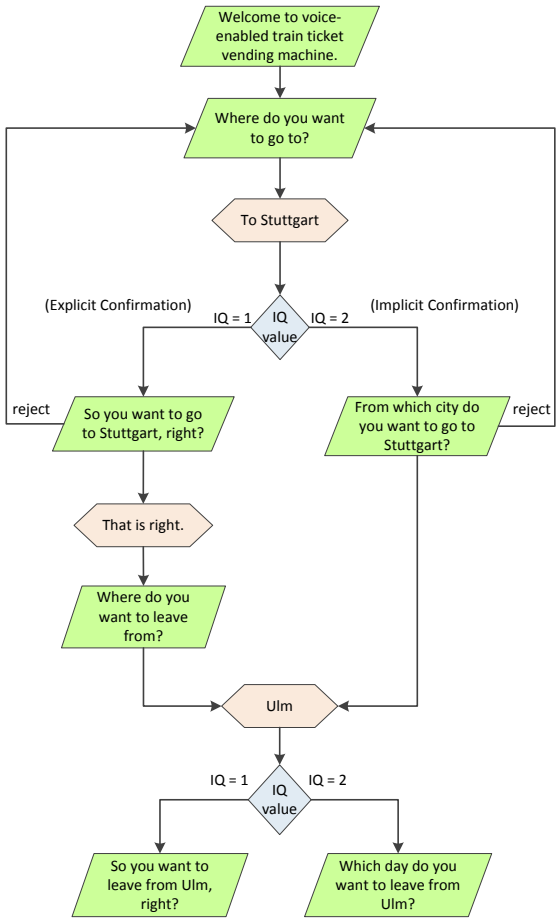


Fig. 5.8: The dialogue flow for the adaptive strategy originally published in (Ultes et al., 2014a): depending on the IQ value, the provided information by the user is either confirmed explicitly or implicitly within the next system question. (Please note: the original dialogue was in German.)

Results

To evaluate the user experiment of adapting the grounding strategy to the user satisfaction, the questionnaires are analysed. The results for each question is depicted in Table 5.1. Each row shows the average score for one of the three different strategies. It is a well known fact that, for simple tasks like this, an all-implicit strategy is usually preferred over an all-explicit strategy (Fraser (cf. 1994)). Hence, as expected, the all-implicit strategy performed best outperforming the all-explicit strategy clearly: it achieved a better score for almost all questions. The difference is even significant for 16 out of 25 values ($\alpha < .05$ applying the Mann-Whitney U test (Mann and Whitney, 1947)). Comparing the all-explicit to the adapted strategy gives a similar impression: The scores for almost all questions are better for the adapted strategy. However, this is not as significant having only 7 significant different values. More revealing is the conclusion drawn from comparing the all-implicit with the adapted strategy. While the all-implicit strategy again governs the scores, almost all results are not significantly different. Hence and in contrast to the

Table 5.1: The average results of the user questionnaires originally published in (Ultes et al., 2014b): each question could be answered by a 7-point scale being translated to scores from one to seven. Significant differences are marked with a, e, and i marking significance with the adaptive, explicit, and implicit strategy respectively. (Please note: the original questionnaire was in German.)

	fun with interaction	system reaction	hard to lose track	natural communication	user is in control	overall impression	user felt calm	user felt tense	dialogue length appropriate	user knew what to say	high level of concentration necessary	system understands user input	functionality	interaction was pleasant	interaction was not exhausting	interaction was not boring	interaction was diversified	interaction was not frustrating	system made no errors	error due to user behaviour	system is friendly	system is easy to use	system is flexible	system is understandable	reaction time	would use system again	human operator preferred	overall satisfaction
all-implicit	5.3^{e}	5.9^{e}	6.2	4.8^{e}	3.0	5.4^{e}	5.0	$6.3^{a,e}$	5.4^{e}	6.0	2.7	5.9^{e}	5.5	5.6^{e}	5.6^{e}	4.7^{e}	3.9^{e}	5.9^{e}	5.9^{e}	3.3	5.8	6.3^{e}	4.1	5.8	5.0^{e}	5.2	4.7	5.5^{e}
all-explicit	$3.7^{a,i}$	4.8^{i}	5.8	3.5^{i}	2.7	4.3^{i}	4.5	5.0^{i}	$3.5^{a,i}$	6.1	3.3	4.2^{i}	4.6	4.3^{i}	4.3^{i}	$3.3^{a,i}$	$3.0^{a,i}$	$4.4^{a,i}$	4.5^{i}	2.1	5.8	5.6^{i}	3.8	5.5	$3.9^{a,i}$	4.2	5.1	$4.4^{a,i}$
adapted	4.8^{e}	5.3	6.1	4.5	3.0	5.0	4.8	5.4^{i}	5.0^{e}	6.2	3.2	5.2	5.3	5.0	5.3	4.3^{e}	3.9^{e}	5.5^{e}	5.3	2.8	5.6	6.0	4.2	5.8	4.9^{e}	5.0	4.7	5.3^{e}

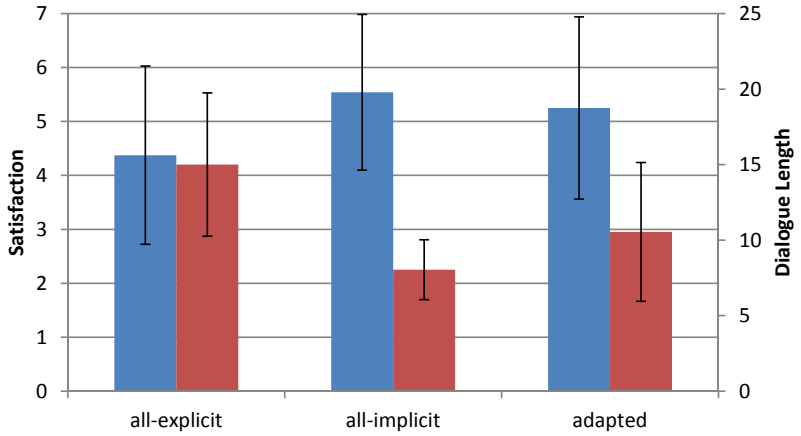


Fig. 5.9: Results of adapting the grounding strategy to IQ originally published in (Ultes et al., 2014b): the overall satisfaction with the dialogue (left bar, left y-axis) and the average dialogue length in number of turns (right bar, right y-axis) according to questionnaire evaluation. Satisfaction for implicit and adapted do not differ significantly while all other differences are significant.

expectations, the adapted strategy did not perform significantly worse despite the dialogue being very simple.

This result is underpinned by looking at the users' overall satisfaction score with the dialogue as an emphasis was put on the question which strategy people liked best. A bar graph showing the average outcome of the user ratings grouped by the respective dialogue strategy is depicted in Figure 5.9. While the adapted strategy resulted in 45.6 % explicit and 54.4 % implicit confirmations, it is very interesting that it was not rated significantly different compared to the all-implicit strategy. That is even, although the ASR component made almost no errors (due to the limited number of options). Moreover, calculating Spearman's Rho (Sec. 2.2.3) shows significant correlation ($\alpha < 0.01$) with $\rho = 0.6$ between the users' overall satisfaction of the all-implicit and adapted strategy. Additionally, the dialogue length, which is one main indicator for user satisfac-

tion in simple dialogues like this, is significantly higher for the adapted strategy compared to the all-implicit strategy.

In other words, although the task was quite simple, there was no difference between the all-implicit and adapted strategies already revealing the potential of quality-adaptation and spurring the hope that for more complex dialogues, a quality-adaptive strategy will perform best.

Building upon these promising results, we have conducted further experiments for adapting a different dialogue strategy aspect: the dialogue initiative. All details of this experiment will be described in the next section.

5.3.2 Adapting the Initiative

While we were clearly able to show encouraging results for adapting the grounding strategy to user satisfaction with the user experiment presented above, it is unclear if other aspects of a dialogue strategy may also be positively affected. Hence, in this contribution, we investigate if applying rules for adapting the dialogue initiative to the user satisfaction—again represented by the Interaction Quality (IQ)—may also result in an increase in IQ and if other metrics like task success rate or dialogue completion rate may correlate[1].

To investigate this, we have designed an experiment having an IQ-adaptive dialogue strategy adapting the dialogue initiative. Depending on the IQ score, the system chooses between three initiative categories. Here, conventional dialogue initiative categories are *user initiative*, *system initiative*, and *mixed initiative* (Sec. 2.1.3). As there are different interpretations of what these initiative categories mean, we stick to the understanding of initiative as used by Litman and Pan (2002): the initiative influences the openness of the system question and the set of allowed user responses. The latter is realised by defining which slot values provided by the user are processed by the system and which ones are discarded. Hence, for *user initiative*, the system asks an open question allowing the user to respond with information for any slot. For *mixed initiative*, the system poses a question directly addressing a slot. However, the user may still provide information for any slot. This is in contrast to the *system initiative*, where the user may only respond with the slot addressed by the system. For instance, if the system asks for the arrival place and the user responds with a destination place, this information may either be used (*mixed initiative*) or discarded (*system initiative*).

Design and Setup

For evaluating rule-based adaptation of the dialogue initiative, five different strategies are created. Three basic non-adaptive strategies are compared against one adaptive and one random adaptive strategy. All of these strategies can be generated from the flow diagram in Figure 5.10 by varying the IQ value. In order to keep the strategies comparable, all have a similar structure: in each strategy, the system starts with an open request allowing the user to respond with information for all slots. The system first continues with confirming provided information before continuing in a strategy-specific way.

For adapting the initiative based on IQ, the following strategies have been used:

[1] Automatic optimisation aims at maximising a reward function. If IQ was contributing positively to this reward function, optimisation would naturally result in an increase in IQ. As we do not perform optimisation within this experiment, this correlation does not automatically exist

User initiative strategy — The initiative is completely on the user's side. Hence, the system continues openly requesting information. If information is provided for any slot, a confirmation is requested from the user. After that, the user response is restricted: new values for already confirmed slots are discarded. This continues until all mandatory slots have a confirmed value or the user terminates the interaction.

Mixed initiative strategy — Having a mixed initiative means that the system is in control of the interaction in general while leaving the user room to change the course of the dialogue proactively. To realise this, the system continues by asking the user to provide information about specific slots. However, the user is not restricted on giving information about the given slot but is free to provide information for any slot. Again, if information is provided, a confirmation is requested from the user restricting the slots the user may refer to in future exchanges.

System initiative strategy — In a strategy based on system initiative, the system is totally in control of the interaction. Thus, the system asks for specific slots discarding information about other slots the user might provide. Hence, the dialogue is less flexible. For provided slot information, a confirmation action is immediately performed.

Adaptive strategy — For adapting the initiative based on IQ, the strategy utilises the basic concepts of the non-adaptive strategies. Hence, the way missing information is requested depends on the Interaction Quality. For an IQ value of five, an open request is placed. For an IQ value greater than two, information for all missing slots is allowed as user input (same behaviour as in the mixed initiative strategy) while only the requested information is allowed otherwise. Again, if unconfirmed slot information is present, the strategy commands to first initialise grounding before requesting missing information. It should be noted that the thresholds between the different adaptation levels have been defined manually based on human judgement. An example dialogue is depicted in Figure 5.11.

Random strategy — The randomly adaptive strategy uses the same dialogue description as the adaptive strategy. However, not the IQ value is used to select the initiative. Instead, the initiative is selected randomly.

The dialogues of all strategies continue until all mandatory slots contain a confirmed value or the user terminates the interaction. If the user responds with information about a slot which is not in the set of allowed slot information, these values are discarded. This may lead to a 'Non-Understanding' (or 'out-of-grammar' user input) even though the user has provided information.

For creating the user-adaptive dialogue, the Let's go bus information domain has been chosen (cf.4.1.1). For creating the user simulator LGUS, Lee and Eskenazi (2012) have reduced the set of system and user actions.

The system actions are as follows:

Request — With using request actions, the system requests information from the user (simulator). Possible requests are 'Request(Open)', 'Request(Bus number)', 'Request(Departure place)', 'Request(Arrival place)', and 'Request(Travel time)'. While the latter four di-

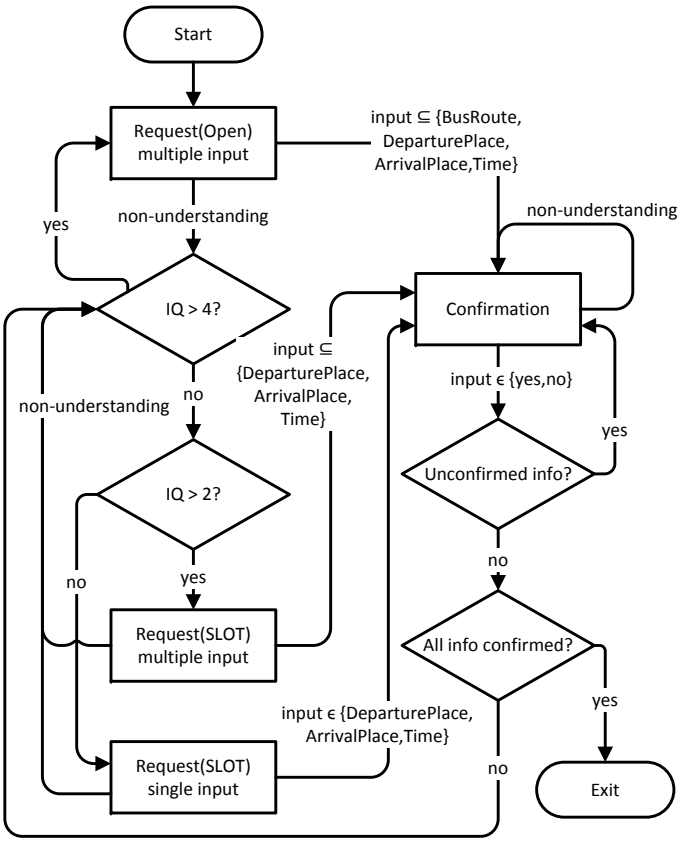


Fig. 5.10: The flow chart describing the adaptive and non-adaptive initiative selection strategies originally published in (Ultes et al., 2015a): for the adaptive strategy, the course of the dialogue as well as the allowed user input are influenced by the IQ value. For the random strategy, the IQ values are generated randomly. The non-adaptive strategies are realised by fixed IQ values: $IQ = 5$ for the user initiative strategy always posing open requests, $IQ = 3$ and $IQ = 1$ for mixed and system initiative explicitly requesting slot information. Provision of the bus route was not mandatory.

rectly target a slot, the first request leaves the choice completely to the user which information to provide to the dialogue system.

Confirm With confirmation actions, the system elicits common ground between the system and the user. Here, the Let's Go User Simulator only provides asking for confirmation explicitly for one slot at a time. Hence, possible actions are 'Confirm(Bus number:VALUE)', 'Confirm(Departure place:VALUE)', 'Confirm(Arrival place:VALUE)', and 'Confirm(Travel time:VALUE)', where VALUE represents the slot value to be confirmed (or denied) by the user.

It is important to note that, in contrast to the number of user actions provided by the user simulator, only one system action at a time may be proposed to the user simulator.

In contrast to this, the user is free to respond with an arbitrary number of user actions at one time (out of the set of available user action). The user actions are:

(1)	S:	Request(Open)	
(2)	U:	Non-understanding	$IQ = 5$
(3)	S:	Request(Open)	
(4)	U:	Inform(Travel Time: 8pm)	$IQ = 5$
(5)	S:	Confirm(Travel Time: 8pm)	
(6)	U:	Deny	$IQ = 3$
(7)	S:	Request(Departure place)	
(8)	U:	Inform(Travel Time: now)	$IQ = 3$
(9)	S:	Confirm(Travel Time: now)	
		⋮	

Fig. 5.11: Example dialogue of the initiative adaptive strategy originally published in (Ultes et al., 2015a): as the IQ value is 5 in the beginning, the system requests openly for information. After the IQ value has dropped to 3, the mixed initiative is active. Hence, the system asks for specific information directly still allowing input for other slots.

Inform For providing new information to the dialogue system, the user informs the system about specific slot values. Hence, the corresponding user actions for informing the system are 'Inform(Bus number:VALUE)', 'Inform(Departure place:VALUE)', 'Inform(Arrival place:VALUE)', and 'Inform(Travel time:VALUE)'. VALUE represents the specific slot value.

Confirm When being ask to confirm the value of a given slot, the user may respond either with an 'Affirm' or a 'Deny' action.

Any combination of the user actions is possible—even having contradicting information present, e.g., informing about two different values of the same slot or affirming and denying a value at the same time. As problems with the speech recognition and language understanding modules are also modelled by LGUS, these effects are reflected by the user action 'Non-Understanding'.

Besides the system and user action, there is also a set of commands for communicating with the user simulator. The command 'Start over' causes LGUS to restart, i.e., creating a new user goal, etc. To get the true user goal from LGUS, which is necessary for evaluating the system by calculating task success, the command 'Get user goal' is provided.

In order to evaluate the dialogue strategies, we use the adaptive dialogue manager OwlSpeak extended for including quality-adaptivity (Sec. 5.2). As OwlSpeak is based on the Model-View-Presenter paradigm, originally, the view is implemented as interface to a voice browser using VoiceXML (Oshry et al., 2007). For connecting OwlSpeak to a user simulator, the view has been replaced. The user simulator at hand, the LGUS (Lets Go User Simulator), is instantiated as a server application communicating to other modules using JSON (Crockford, 2006). Furthermore, the system has been extended to handle multi-slot user input.

For rendering the system adaptive, the interaction estimation module is used as presented in Section 5.2.4. Interaction with real users requires a more complex system than an interaction with a simulated user. Thus, some SDS modules are missing and not all parameters of the Interaction Quality paradigm are available and may be used. This results in a feature set of only 16

parameters[2]. The trained model achieves an unweighted average recall[3] of 0.55 on the training data using 10-fold cross-validation which is comparable to our best-know approaches presented in Section 4.1. All exchanges of the *LEGO* corpus have been used for training.

Evaluation of the dialogue strategies is performed by creating 5,000 simulated dialogues for each strategy. Like Raux et al. (2006), short dialogues (less than 5 exchanges[4]) which are considered "not [to] be genuine attempts at using the system" are excluded from all statistics of this experiment.

In addition to the three objective metrics average dialogue length (ADL), dialogue completion rate (DCR) and task success rate (TSR) described in Section 2.2.3, which are used to evaluate the dialogue performance, the average IQ value (AIQ) is also considered. It is used to investigate a correlation between objective measures and IQ. AIQ is calculated for each strategy based on the IQ values of the last exchanges of each dialogue. Furthermore, this measure is also used to investigate if adapting the course of the dialogue to IQ also results in higher IQ values.

Results

Figure 5.12 shows the ratio of complete, incomplete, and omitted dialogues for each strategy with respect to the total 5,000 dialogues. As can be seen, about the same ratio of dialogues is omitted due to being too short. The DCR clearly varies more strongly for the five strategies.

The results for DCR, TSR, ADL, and AIQ are presented in Table 5.2 and Figure 5.14. TSR is almost the same for all strategies, meaning that, if a dialogue completes, the system almost always found the correct user goal.

Table 5.2: The results of the experiments for the five strategies given by dialogue completion rate (DCR), task success rate (TSR), average dialogue length (ADL) and average Interaction Quality (AIQ) rating the complete interaction for all completed dialogues originally published in (Ultes et al., 2015a). All results for DCR and TSR are significantly different (chi-squared test). Significant differences in ADL (unpaired t-test) and AIQ (Mann-Whitney U test) with the respective column below are marked with ** for the level of $\alpha < 0.01$ and with * for $\alpha < 0.05$. All other comparisons between non-neighbours are significant with $\alpha < 0.01$

Strategy	DCR	TSR	ADL	AIQ
adaptive	54.27%	99.18%	11.86	3.47^{**}
random	49.53%	99.22%	11.82^{**}	3.44^{**}
system initiative	29.48%	98.75%	13.30^{*}	3.23
mixed initiative	22.91%	99.20%	14.40^{*}	3.15^{**}
user initiative	5.32%	97.92%	18.04	2.66

DCR, ADL and AIQ on the other hand vary strongly. They strongly correlate with a Pearson's correlation of $\rho = -0.953$ (level of significance $\alpha < 0.05$) for DCR and ADL, $\rho = 0.960$ ($\alpha <$

[2] The parameters applied are ASRRecognitionStatus, ASRConfidence, RePrompt?, #Exchanges, ActivityType, Confirmation?, MeanASRConfidence, #ASRSuccess, %ASRSuccess, #ASRRejections, %ASRRejections, {Mean}ASRConfidence, {#}ASRSuccess, {#}ASRRejections, {#}RePrompts, {#}SystemQuestions.

[3] The arithemtic average over all class-wise recalls.

[4] The minimum number of exchanges to successfully complete the dialogue is 5.

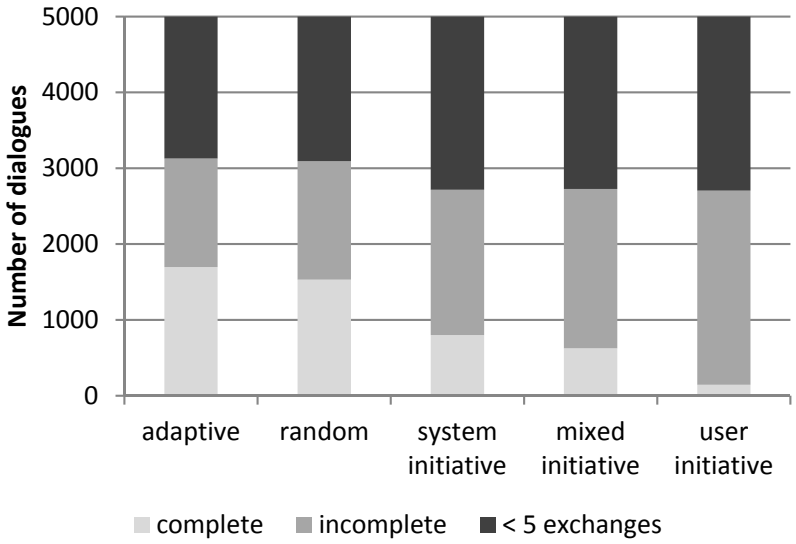


Fig. 5.12: The ratio of omitted dialogues for adapting the initiative originally published in (Ultes et al., 2015a). The bars show the dialogues omitted due to their length (< 5 exchanges), the completed dialogues (complete), and the dialogues which have been aborted by the user (incomplete) with respect to the dialogue strategy. While the amount of short dialogues is similar for each strategy, the number of completed dialogues varies strongly.

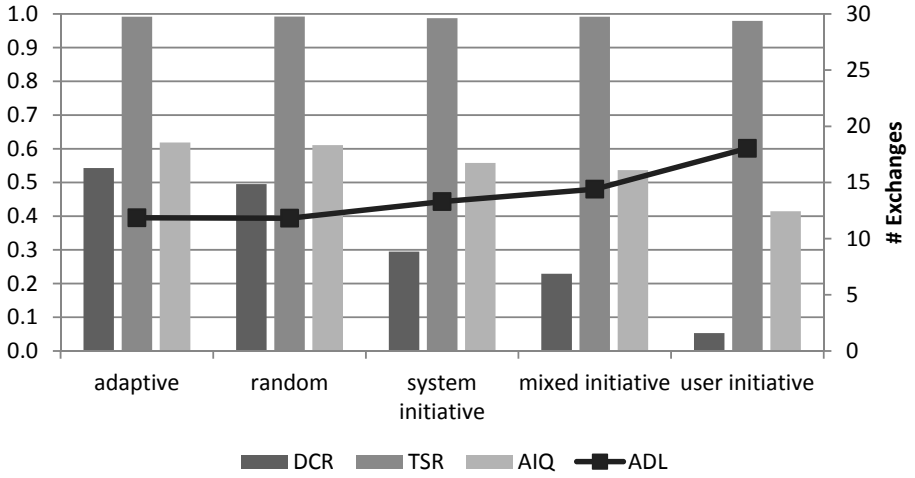


Fig. 5.13: The dialogue completion rate (DCR), the task success rate (TSR), the average Interaction Quality (AIQ), and the average dialogue length (ADL) for all rule-based dialogue strategies originally published in (Ultes et al., 2015a). With decreasing DCR, also AIQ decreases and ADL increases. (AIQ values are normalised to the interval [0,1].)

0.01) for DCR and AIQ, and $\rho = -.997$ ($\alpha < 0.01$) for ADL and AIQ. This shows that by improving IQ, being a subjective measure, an increase in objective measures may be expected.

Comparing the performance of the adaptive strategy to the three non-adaptive strategy clearly shows that the adaptive strategy performs significantly best for all metrics. With a DCR of of 54.27%, the performance is comparable to the rate achieved on the training data of LGUS (cf. (Lee and Eskenazi, 2012)). The non-adaptive strategies achieve a much lower DCR having the system initiative strategy as second best with only 29.48%. This performance goes together with shorter dialogues shown by the ADL. Furthermore, the results for DCR clearly show that the user initiative strategy is unemployable. Thus, this strategy will not be analysed any further.

Furthermore, it is of interest if better objective performance also results in better IQ values for the complete dialogue. This is especially important since it is imperative for the relevance of the Interaction Quality. Adapting to IQ to improve the dialogue should also result in an increase of the IQ value. This effect has been validated by these experiments. The adaptive strategy has a significant higher average IQ (AIQ) value calculated from the IQ value for the whole dialogues, i.e., the IQ value of the last system-user-exchange, than all other non-adaptive strategies.

The question remains if adapting to IQ is the actual reason for the improvement. Maybe, the user only likes diversified initiative prompts better which is represented by the random strategy. While this statement is true to some extent (see ADL), reasonably adapting to IQ further improves the system performance significantly as shown by DCR and AIQ.

Besides the interest in the general performance of the quality-adaptive strategy, we are specifically interested whether implications may be drawn from the experiments about the usage of IQ in a reinforcement learning setting for modelling the reward function which will be presented in the following.

5.3.3 Reward Modelling

For modelling the reward employing the user state, we have outlined two different approaches in Section 5.1.2. One of these approaches describes how to use the user state to decide whether the dialogue was successful or not to apply a high positive or negative reward accordingly. This idea is investigated more closely in this experiment. Here, we start off with an analysis of the results of the previous experiment with respect to the potential of IQ for reward modelling. Based on these findings, a reward function using IQ for defining success is applied for POMDP-based reinforcement learning in a follow-up experiment. The resulting policies are evaluated against policies trained on state-of-the-art approaches to reward modelling.

Analysis: IQ for Reward Modelling

The presented results in Section 5.3.2 clearly show that AIQ and DCR are correlated. As almost all completed dialogues were also successful, a correlation between AIQ and task success may be assumed. In this section, we investigate if this correlation may be exploited for modelling the reward function for reinforcement learning approaches to dialogue management. This would be very beneficial, as for state-of-the-art reinforcement learning approaches to dialogue management, e.g., (Lemon and Pietquin, 2012; Young et al., 2013), a positive or negative reward is added at the end of each dialogue depending on the successful achievement of the task. However, to do this, usually, the true user goal needs to be know. This is either possible by asking the user or by

Table 5.3: Example of the task success rate with respect to IQ and the dialogue length (DL) originally published in (Ultes et al., 2015a). Not regarding rows with less than 15 dialogues, there is clearly a trend for higher task success rates if the IQ value increases as well.

DL	IQ	success	failure	# dialogues
9	1	0.0%	100.0%	487
	2	0.0%	100.0%	40
	3	37.2%	62.9%	253
	4	93.8%	6.3%	512
	5	0.0%	100.0%	2
10	1	0.0%	100.0%	452
	2	0.0%	100.0%	38
	3	42.4%	57.6%	172
	4	96.6%	3.5%	406
	5	0.0%	100.0%	3
11	1	0.0%	100.0%	405
	2	2.9%	97.1%	35
	3	47.8%	52.3%	178
	4	84.0%	16.0%	100
	5	-	-	0
12	1	0.3%	99.7%	329
	2	23.1%	76.9%	52
	3	78.5%	21.6%	297
	4	96.3%	3.7%	270
	5	0.0%	100.0%	1

using a user simulator for training. Here, it has been shown that optimising the strategy with real user dialogues yields better strategies (Gačić et al., 2013a) than using a user simulator. However, asking the user to provide whether they consider the dialogue to be successful is time consuming and interruptive thus only possible in artificial lab settings. If there was a metric which allowed to automatically detect successful, or, more generally, good dialogues, this metric would be very useful for the before described situation yielding the opportunity to optimise on real dialogues without disrupting the users.

Therefore, the correlation of the final IQ value and task success is analysed. Based on all strategies, the dialogues are evaluated regarding the success rate with respect to the final IQ value and the dialogue length. An example for dialogue lengths of 9–12 is depicted in Table 5.3. To compute those, again, dialogues with less than five exchanges are excluded. Clearly, a trend can be identified for higher task success rates when having a high final IQ for all dialogue lengths[5].

Based on this finding, an IQ threshold may be defined which separates dialogues regarded as being successful and dialogues regarded as being not successful. For a threshold of four, for example, all dialogues with a final IQ of five and four may be regarded as successful while all other dialogues are regarded as failure. However, not all dialogues above the threshold are necessarily actually successful and not all dialogues below the threshold are necessarily actually un-

[5] Only rows with more than 15 dialogues are regarded as sufficient data is needed to compute reasonable task success rates.

Table 5.4: The precision of success and failure dialogues (along with the unweighted average precision (UAP)) when setting all dialogue with final IQ greater or equal a given IQ value to be successful and the remainder to be a failure originally published in (Ultes et al., 2015a).

Success	Precision		UAP
IQ $\geq$	Success	Failure	
5	0.448	0.669	0.559
4	0.863	0.826	0.845
3	0.652	0.888	0.770
2	0.646	0.995	0.820
1	0.331	-	0.166

successful. Hence, to find an optimal threshold, the precision—representing this relationship—is calculated for both success and failure dialogues for different thresholds. The results are depicted in Table 5.4.

The best overall threshold indicated by a maximum unweighted average precision[6] (UAP) is four achieving a precision of 0.863 for success and of 0.826 for failure. While a threshold of four is also the best threshold for success, the highest precision for failure is a threshold of two, i.e., regarding all dialogues as being a failure with a final IQ of one. Hence, to further maximise UAP, two thresholds may be defined: four for success and two for failure. This results in an UAP of 0.929 not regarding all dialogues with a final IQ of two or three.

Defining a threshold based on precision yields the downside that some actually successful dialogues are regarded as failure and vice versa. In fact, defining a threshold of four results in a recall—representing the percentage of dialogues being regarded as successful out of all truly successful dialogues—of 0.595 as shown in Table 5.5. This means that more than 40% of all truely successful dialogues are regarded as failure which is not ideal. Additionally, a recall of 0.953 for failure means that less than 5% of all truly failing dialogues are regarded as success. However, using the two thresholds defined above results in better rates. Still, 4.7% of all failing dialogues are regarded as success. However, only 0.8% of all successful dialogues are regarded as failure which is much better. Having two thresholds, though, results in the need for more training dialogues: only 64% of all dialogues are used for training resulting in the need for 56% more dialogues for training.

Design and Setup

While we have above argued for using the IQ value within a reward function, the proposed methods still have to be validated. Hence, we have designed an experiment where we use the IQ value for deciding whether a dialogue is regarded as being successful. More precisely, we follow the above findings by defining each dialogue with a final IQ value of ≥ 4 as successful. This reward function is evaluated against a reward function utilising true task success instead. Both reward functions have in common that, for each turn, -1 is added to the total reward R of the respective dialogue. In the end of each dialogue, a decision is made whether a reward of $+20$ is added to R. Here, the following reward functions have been applied for calculating R:

[6] The arithmetic average over all class-wise precisions.

Table 5.5: The recall of success and failure dialogues (along with the unweighted average recall (UAR)) when setting all dialogue with final IQ greater or equal a given IQ value to be successful and the remainder to be a failure originally published in (Ultes et al., 2015a).

Success	Recall		UAR
IQ $\geq$	Success	Failure	
5	0.008	0.995	0.502
4	0.595	0.953	0.774
3	0.798	0.789	0.794
2	0.992	0.730	0.861
1	1.000	-	0.500

$$R_{TS} = T \cdot (-1) + \begin{cases} +20 & \text{if task successful ,} \\ 0 & \text{otherwise ,} \end{cases} \tag{5.9}$$

$$R_{IQ} = T \cdot (-1) + \begin{cases} +20 & \text{if dialogue completed \& IQ } \geq 4 \text{ ,} \\ 0 & \text{otherwise .} \end{cases} \tag{5.10}$$

For the experiments, we have used the same dialogue setup as for the rule-based experiment described in Section 5.3.2 and illustrated in Figure 5.8. However, the rules which have been used for selecting the type of dialogue initiative have been removed. Instead, the best dialogue initiative strategy is determined automatically during the reinforcement learning process. Another difference is that, while we have used one state hypothesis for the previous experiment, for this experiment, HIS-OwlSpeak is used having multiple state hypotheses thus constituting a full-blown POMDP. Its implementation in our OwlSpeak dialogue manager is described in Section 5.2.

Applying a Gaussian process-based learning algorithm as presented in Section 2.1.2 for the Let's Go domain also encompasses the definition of a suitable kernel function. The kernels proposed by Gačić and Young (2014) resulted in numerical inconsistencies. Therefore, we have defined our own kernel inspired by Lefevre et al. (2009) as

$$\begin{aligned} k((b,a),(b',a')) = &\sum_{d=1}^{2} \alpha_d \cdot (1 - |b(d) - b'(d)|) \\ &+ \sum_{d=3}^{5} \alpha_d \cdot \delta(b(d), b'(d) \\ &+ \alpha_a \cdot \delta(a,a') \text{ .} \end{aligned} \tag{5.11}$$

Here, the αs are weights and $d \in [1,5]$ represents the dimension of the belief state having $d \in [1,2]$ referring to the continuous values and $d \in [3,5]$ referring to the discrete values of the belief state. $\delta(x,y)$ represents the Kronecker delta which evaluates to 1 only if $x = y$ and 0 otherwise.

As we have proposed different variants of modelling IQ in Section 4.1, we will not only use an SVM within the IQ estimation module (Sec. 5.2.4) but also a Hybrid-HMM (see Sec. 4.1.3) approach using the SVM as classifier for providing the observation probability. Moreover, we will also apply a random generator for generating the IQ value randomly. Consequently, four different reward variants are investigated: R_{TS}, R_{IQ-SVM}, R_{IQ-HMM}, and R_{IQ-RND}. For each reward function, a policy is trained with 1,000 dialogues using the GP-SARSA algorithm (Sec. 2.1.2) and

evaluated with 5,000 simulated dialogues. For training and evaluation, the Let's Go User Simulator (Lee and Eskenazi, 2012) is applied. As in the previous experiment, the evaluation metrics TSR, DCR, ADL, and AIQ are used.

Results

The results for the policies trained using the four proposed reward functions are depicted in Figure 5.7 and Table 5.6. Evidently, the policy trained using R_{IQ-SVM} outperforms all other policies significantly in DCR, ADL, and AIQ. Even the differences to R_{TS}, its closest competitor, are significant (DCR: $p < 0.001$ using chi-squared test, ADL: $p < 0.001$ using unpaired t-test, AIQ: $p < 0.001$ using Mann-Whitney U test). Moreover, achieving a DCR of 60.61%, the learned strategy outperforms all rule-based strategies evaluated in the previous section. When comparing the different ways of recognising IQ, using the SVM results in best performance. Thus, while the Hybrid-HMM achieves a good overall correlation on the exchange-level it is not as suitable for estimating the final IQ value compared to applyinig an SVM. Still, the resulting strategy performs still better than using a randomly generated IQ value.

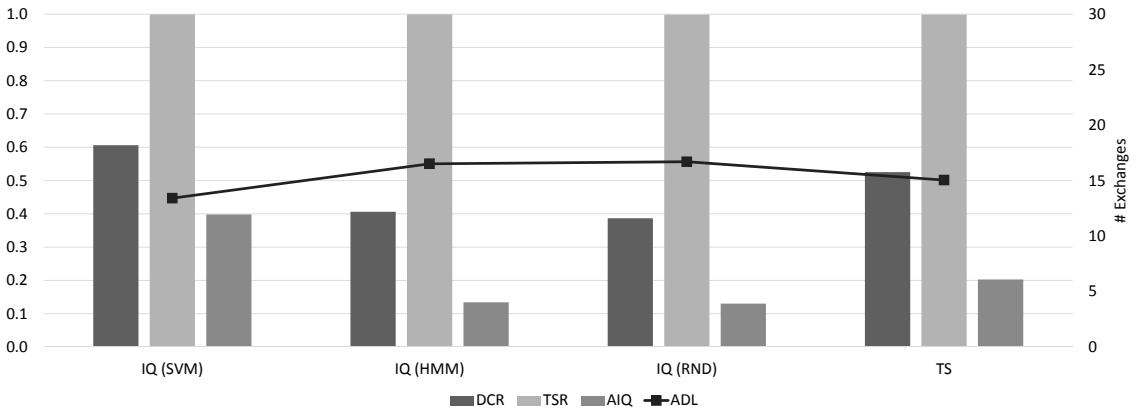


Fig. 5.14: The dialogue completion rate (DCR), the task success rate (TSR), the average Interaction Quality (AIQ), and the average dialogue length (ADL) for all learned policies with HIS dialogue management. Evidently, using the Interaction Quality for deciding which dialogue was successful achieved best performance in all measures. (AIQ values are normalised to the interval [0,1].)

Of most interest, though, is the difference between R_{IQ-SVM} and R_{TS}. As almost all dialogues which are completed are also successful, using R_{IQ-SVM} may be seen as an additional selection process: out of all successful dialogues, only the ones with a high IQ value are selected as positive examples for the learning process. The distribution of dialogues with a final $IQ \geq 4$ and $IQ < 4$ for all successful dialogues during the learning process using R_{IQ-SVM} is depicted in Table 5.7. Only 31.3% of the successful dialogues are actually used as positive examples. Hence, it may be concluded that IQ encapsulates more information about positive dialogues than a simple measure like task success is able to. This finding has also been made in a similar way by Gačić et al. (2013a) using the rating of real users acquired through human interaction.

Table 5.6: The results of the POMDP-based strategies using the different reward functions based on task success (TS) or the final IQ value. The latter has been determined using either an SVM or an HMM, or a random generator. The strategy which used the IQ value of SVM-based recognition performs best.

	DCR	TSR	AIQ	ADL
IQ (SVM)	60.61%	99.90%	2.59	13.41
IQ (HMM)	40.62%	99.93%	1.54	16.51
IQ (RND)	38.66%	99.84%	1.52	16.70
TS	52.53%	99.89%	1.81	15.04

Table 5.7: Distribution of IQ values with respect to task success for the training using the IQ-based reward function IQ (SVM). Only 31.3% of all truly successful dialogues have an IQ value which was high enough to result in a high final reward.

IQ	percentage
≥ 4	31.1%
< 4	68.9%

Up until now, we have presented the evaluation of adapting the course of the dialogue according to the user satisfaction for rule-based systems. Furthermore, we have outlined and evaluated the usability of user satisfaction for learning an optimised policy. For evaluating adaptation on perceived coherence, we have chosen our approach on re-ranking system actions. This will be described in detail in the following.

5.4 Experiments and Evaluation of Perceived Coherence Adaptation

Having the user satisfaction as the most universal user state (out of all four presented user states in Chapter 4), the perceived coherence clearly resides at the second position in our user state ranking: it is comparable to the user satisfaction in terms of application generality and provides almost the same potential on improving the dialogue. Hence, we present an experiment of adapting the course of the dialogue to the perceived coherence. Out of our proposed adaptation methods presented in Section 2.1.3, we have selected the re-ranking of dialogue acts as appropriate approach for our experiment which will be presented in the following.

5.4.1 Re-ranking of System Actions

As mentioned before, one important aspect of human-machine-communication poses the problem of generating coherent system responses, i.e., the system response should be coherent with the conversation up to the current moment. Here, utilising the coherence information extracted from the conversation may help. In order to do so, our idea has been outlined in Section 5.1.3: by estimating the perceived coherence of system dialogue acts (SDAs) of an n-best-list provided

by the actual dialogue manager, the overall coherence of the dialogue system is improved by selecting the SDA with highest coherence. The evaluation of our approach will be described in this section starting with the experiment's design and setup.

Design and Setup

A suitable setup was chosen for evaluation which is depicted in Figure 5.15. Here, the dialogue manager, which is applied within the first step in our two-step process, generates the n-best-list and is based on the approach of Griol et al. (2008). They created a statistical dialogue manager employing supervised learning techniques. Based on a labeled corpus, a dialogue register as described in Section 4.2.3 is used modelling the current dialogue state. Utilising a multi-layer perceptron (MLP, see Sec. 2.2.1), a statistical model is trained determining the next system action taking the current dialogue state, i.e., the dialogue register, as input. As we use the dialogue manager not only to determine the one-best system action but to determine the n-best system actions, this list is generated using the probabilities of each possible system action computed by the MLP.

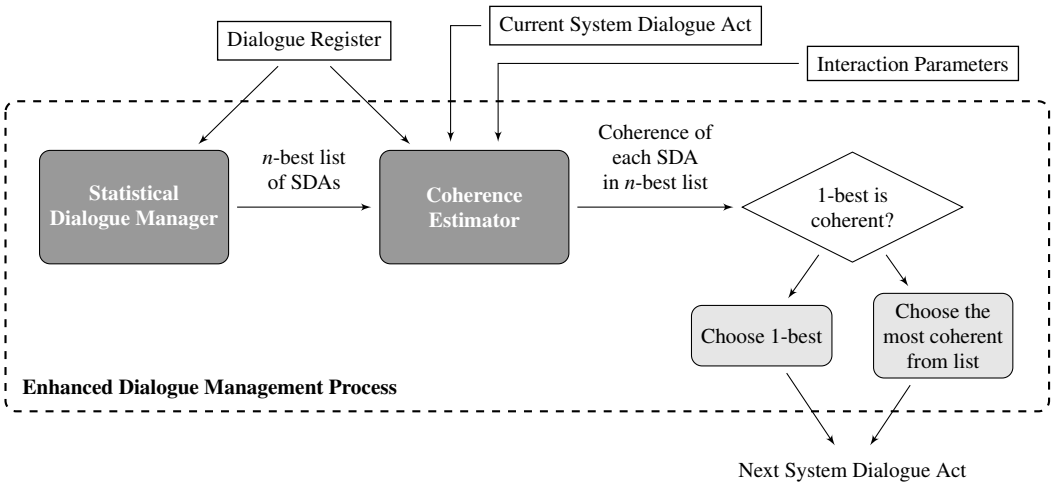


Fig. 5.15: Enhanced dialogue management process for a two-stage coherence-based system action selection. The *n*-best list of system dialogue acts (SDAs) is tested for coherence. If the 1-best entry is coherent, then it is selected. Otherwise, the most coherent alternative from the list is chosen.

Having an n-best-list of system dialogue act generated from the dialogue manager, each system dialogue act is then evaluated whether being coherent or not given the conversation up to the current turn. More precisely, only the top-ranked SDA is tested, which is the one conventionally being executed. If this SDA is predicted to be coherent, it is directly executed[7]. Only if the top-ranked SDA is predicted to be non-coherent, the remaining SDAs of the n-best-list are tested for coherence. Based on the results, the n-best-list is re-ranked to have only coherent SDAs on top. If there are multiple SDAs being predicted as coherent, the confidence scores from the estima-

[7] This is even if other SDAs may be evaluated to be coherent with a higher confidence. The selection of this particular SDA may then be an effect of the applied dialogue strategy.

Table 5.8: Experimental results (absolute and relative) of coherence-based re-ranking of system dialogue acts compared to a plain statistical dialogue manager without re-ranking as well as the original system.

	Re-ranked	Non re-ranked	Original LetsGo
Non-coherent	685 (16.2%)	717 (17.0%)	395 (9.4%)
Weakly coherent	876 (20.7%)	889 (21.1%)	1005 (23.8%)
Strongly coherent	2661 (63.0%)	2616 (62.0%)	2822 (66.8%)

tor, which are associated with the prediction quality, are also regarded having predictions with a higher confidence on top of the list.

To evaluate this approach, the proposed method as been applied to compute an alternative option from a 4-best list. For the initial test for coherence, a binary classifier was trained, i.e., the classes of strongly and weakly coherent were combined to the class coherent. The resulting binary classifier achieved an unweighted average recall of 62.2% based on 5-fold cross validation. If this initial estimation resulted in a coherent SDA, then the SDA is executed directly. Otherwise, all candidates of the 4-best-list are tested for coherence using our approach presented in Section 4.2 and the list is re-ranked.

Results

The resulting dialogue acts are again annotated with coherence allowing us to compute an overall coherence rate. Table 5.8 shows the experimental results comparing the annotated coherence of the responses selected by the original rule-based Lets Go system, the statistical dialogue manager developed, and the statistical dialogue manager optimised by means of the proposed technique.

As can be observed, the statistical dialogue manager developed achieves coherence rates lower than the original LetsGo system. That is, the original system—which is based on fine-tuned hand-crafted rules—selects SDAs which are more coherent. One issue is that the training data of the statistical dialogue manager, which is based on the *LEGO* corpus (see Sec. 4.1.5), does not provide coherent SDAs for all situations. This includes unseen situations as well as situations where the original system chose a non-coherent SDA.

However, using the proposed technique, the statistical dialogue manager could be improved by using the coherence estimator. Many of the non-coherent 1-best dialogue acts selected in the first place were corrected, the number of non-coherent DAs could be reduced by 0.8% absolute (4.45% relative) and increasing the number of strongly coherent SDAs by 1.0% absolute at the same time.

For dialogue systems in a more complex domain, it may be expected that the impact of coherence-based re-ranking is even higher. For such a system, it is quite hard to define a well-performing set of rules. Thus, it may be expected that the overall coherence is also lower. Furthermore, systems using statistical methods for creating policies have shown to perform better in complex domains. Due to the nature of our proposed system to not be dependent on the actual dialogue management approach, adding coherence-based re-ranking of SDAs may be considered as being able to improve the DM output in general.

For a better understanding, an example dialogue is shown in Figure 5.16 showing a dialogue where the SDM chooses a non-coherent system action. However, by re-ranking the SDM list of SDAs based on coherence, a coherent dialogue act is chosen.

		Utterance	
(1)	S:	Welcome to the CMU Let's Go bus information system. What can I do for you?	(SDA Welcome)
(2)	U:	CAN USE 16B FROM TELL ME TO DOWNTOWN PITTSBURGH	
(3)	S:	Going to DOWNTOWN PITTSBURGH. Is this correct?	(SDA Confirm_destination)
(4)	U:	YES	
(5)	S:	Okay. Where do you wanna leave from?	(SDA Ask_for_origin)
(6)	U:	DOWN YES	
(7)	S:	For example, you can say, FORBES AND MURRAY, DOWNTOWN, or MCKEESPORT.	(SDA Ask_for_origin)
(8)	U:	TOWN YES	
(9a)	S:	*Non re-ranked*: —	(SDA Confirm_origin)
(9b)	S:	*Re-ranked*: Which neighbourhood do you want to leave from?	(SDA Ask_for_origin_neighbourhood)

Fig. 5.16: A dialogue example showing the difference of using SDM with and without coherence-based re-ranking: at the end of the dialog, the former selects the non-coherent SDA *Confirm_origin*. By re-ranking, the SDA *Ask_for_origin_neighbourhood* is selected instead which is strongly coherent.

5.5 Conclusion on User-Adaptive Dialogue Management

With the general goal of rendering the dialogue manager user-adaptive, we have presented three approaches of introducing user state information into the dialogue management process: a rule-based approach, an approach based on policy optimisation and an approach on selecting the system action according to the prospective change in coherence.

For rule-based adaptation to user satisfaction, a pilot experiment has been conducted with real users. In the study, we investigated dialogues whose confirmation strategy was adapted to the user satisfaction represented by Interaction Quality (IQ). We could show that, even for simple dialogues, the adaptive strategy was not significantly worse than an all-implicit strategy which is known to work best for simple dialogues like the one applied. Therefore, we believe that for more complex dialogues an adaptive strategy will perform best. For applying a similar approach to adapting the dialogue initiative to the user satisfaction, we further analysed the performance of an adaptive dialogue strategy. Furthermore, we were able to shed light on the question if IQ and objective measures correlate in such a setting. By comparing five different strategies, we could show that the dialogue completion rate, the average dialogue length, and the average Interaction Quality strongly correlate. In addition, we could show that the adaptive strategy clearly outperforms all non-adaptive strategies as well as the random strategy. Hence, not only the grounding strategy but also the dialogue initiative is suitable for rule-based quality-adaptive dialogue.

By performing a more detailed analysis of the correlation of task success and IQ within the experiment on adapting the initiative to IQ, we were able to show that defining IQ thresholds sepa-

rating dialogues regarded as success and failure is a reasonable approach achieving an unweighted average precision of 0.845 for defining one threshold or 0.929 for defining two thresholds. This is of special interest for reinforcement learning where this may be used to automatically detect task success. In fact, modelling the reward accordingly using one threshold results in better performance compared to using task success itself as shown in our experiment. Here, IQ functions as an additional filter which is able to select positive dialogue examples better than task success. While no experiments have been conducted for the case of having two thresholds, it should still be noted that it remains unclear what to do with the dialogues which result in a final IQ between the thresholds. If those dialogues are not used for training, this would result in the need for 56% more training dialogues.

Finally, we also conducted an experiment for our novel approach on using coherence information in the dialogue manager. To evaluate our proposal we have annotated a corpus of 200 dialogs with 4,222 system-user exchanges. The experimental results show that although the dialogue manager implemented obtained similar coherence rates than the original handcrafted Let's Go system, our technique made it possible to improve 4.46% of the DAs selected, which were replaced by more coherent alternatives.

6

Conclusion and Future Directions

In this thesis entitled "User-centered Adaptive Spoken Dialogue Modelling", we presented work on rendering the speech interaction between a machine and a human more user-centred by introducing automatic user state recognition into the dialogue management process. More specifically, we divided the overall problem into two sub-problems and addressed them separately: (i) recognising the user state using statistical models and (ii) adapting the course of the dialogue accordingly.

For user state recognition, we investigated the four user states *user satisfaction*, *perceived coherence*, *emotion*, and *intoxication* and were able to provide work which advances the state of the art for all four user states. For user satisfaction, we were especially interested in adding a temporal context to its recognition approaches which are based on statistical classification by using Markovian approaches or a modified feature set. Here, we were able to prove our hypothesis by showing that temporal information has a major influence on the recognition performance. An optimised feature set which reflected the temporal information provided the highest overall significant improvement compared to the state-of-the-art baseline achieving an UAR of 0.549 (+15.69%), a κ of 0.679 (+12.42%), and a ρ of 0.812 (+7.13%).

From our work on recognising the perceived coherence, we were able to reveal that there exists a relationship between events of the interaction and the coherence. This has been demonstrated by creating a classification approach using interaction parameters together with the dialogue state and the system action resulting in an UAR of 0.623. Furthermore, we disclosed that the performance of speech-based emotion recognition may be improved by taking the speaker into account and creating speaker-dependent recognition models. Here, we were able to improve the recognition rate significantly by up to +9.42% compared to speaker-independent state-of-the-art approaches.

With our work on user-adaptive dialogue management, an emphasis was placed on adapting the course of the dialogue to user satisfaction. Here, we achieved promising results: adapting basic strategy aspects of the dialogue (the initiative or the confirmation prompts) may lead to an increase in user satisfaction. This was demonstrated in three experiments. For adapting the grounding strategy in a user experiment, the adaptive strategy selecting the confirmation prompt based on the user satisfaction was under the two best performing strategies. For adapting the initiative, we showed that the adaptive strategy significantly outperformed all other strategies in terms of dialogue completion rate (54.27%), average dialogue length (11.86), and average Interaction Quality (3.47) compared to non-adaptive state-of-the-art strategies. Additionally, we were

able to identify a correlation between this improvement in user satisfaction and an improvement of other objective performance measures like completion rate or average dialogue length. Thus, it may be expected that for adapting to more complex aspects of the strategy, the influence of the user satisfaction may even be higher. Furthermore, we analysed the usability of user satisfaction and showed the potential for modelling the reward function of reinforcement learning based dialogue management. By applying our proposed approach, 92.9% of all dialogues regarded as success or failure were actually successful of unsuccessful, respectively. In the following experiment using our POMDP dialogue manager implementation comparing a reward function based on task success with our proposed method of using the Interaction Quality value, the latter achieved the best dialogue completion rate of 60.61%.

Finally, we proposed a novel approach on incorporating the coherence into the dialogue management by re-ranking an n-best-list of system actions provided by a dialogue manager according to their perceived coherence and selecting the top coherent action. This approach has resulted in a significant improvement in a simulated dialogue management task by replacing 4.46% of all non-coherent system actions with more coherent ones.

While we have only outlined the major findings of this thesis, a more detailed description of our contributions will be presented in the following.

6.1 Thesis Contributions

During our work on user-centred adaptive spoken dialogue systems which has been presented in this thesis, we have achieved several contributions advancing the state of the art. Those may be grouped into *theoretical*, *practical* and *experimental* contributions and will be described in this section.

6.1.1 Theoretical

For our work on automatically identifying the user state, we formulated the problem of recognising the user satisfaction as a sequential problem using a Hidden Markov Model and a Conditioned Hidden Markov Model (Ultes et al., 2012a,b). In this context, we created two novel approaches for recognising the user satisfaction in both a static and a sequential setting. By introducing the *error correction approach* thus automatically identifying the error of a standard static machine learning approaches the hypotheses may be corrected accordingly (Ultes and Minker, 2013b). Furthermore, modelling the observation probability of a Hidden Markov Model using static classifiers introduces temporal dependencies into the recognition process resulting in a *Hybrid Hidden Markov Model* (Ultes and Minker, 2014a).

By connecting interaction-related events with the perceived coherence of the system reaction, we were the first to formulate the problem of predicting the coherence as a static recognition task. For emotion recognition, we were the first to propose two *speaker-dependent emotion recognition* approaches using information about the speaker within the classification model (Sidorov et al., 2014b,a).

To utilise the user state information within the dialogue management, we formulated a rule-based approach and also built upon partially observable Markov decision processes (POMDPs) for modelling uncertainty inherent in dialogue systems (Ultes et al., 2012c). For the latter, we

extended the formalism to include user state variables and adapt all equations accordingly (Ultes et al., 2011a). By this, a strategy optimised by reinforcement learning may also take into account the user state information. Additionally, we proposed a *reward modelling* strategy which may be employed to automatically finding the optimal policy using reinforcement learning (Ultes et al., 2015a). In contrast to conventional reward modelling strategies, the (current) user state is utilised for determining the reward during or at the end of the dialogue. Furthermore, a novel architecture for dialogue management has been developed. This architecture allows to *select the next system action by predicting the change* the system action may have on the user state.

6.1.2 Practical

For our experiments, the above described theoretical contributions have been implemented. For recognising all user states, we implemented the investigated approaches using off-the-shelve algorithms and frameworks. Furthermore, we have created an open source Java library implementing a Conditioned Hidden Markov Model (Ultes et al., 2013a).

Within the context of user state recognition, we have created and published a complete corpus for user satisfaction as well as for perceived coherence (Schmitt et al., 2012; Ultes et al., 2015b).

Evaluating adaptive dialogue modelling requires the integration of the proposed techniques into an existing dialogue manager. Here, we re-engineered an already existing dialogue manager by adding a module for automatically recognising the user state at hand (Ultes et al., 2014a). To conduct experiments with a POMDP, the rule-based dialogue manager has partially been redesigned to include the POMDP functionality which also entailed the extension of the POMDP formalism with the user state (Ultes and Minker, 2013a, 2014b).

6.1.3 Experimental

With the proposed theories and practical implementations, several experiments for user state recognition have been conducted. By focusing on temporal aspects for user satisfaction recognition, i.e., taking into account dependencies of the current dialogue situation on the complete dialogue, we have provided evidence to conclude that these temporal aspects are of great importance. For this, experiments with all proposed static and sequential algorithms have been executed in various settings (Ultes et al., 2012a,b, 2013b; Ultes and Minker, 2013b,c, 2014a; Schmitt and Ultes, 2015). Here, we were able to improve the baseline by up to 13.23%. For our experiments on coherence prediction, we achieved a performance of 0.623 UAR. Within our experiments on speaker-dependent emotion recognition, we were also able to show that personalised models yield better performance on different corpora increasing the recognition performance by up to +9.42% (Sidorov et al., 2014b). Finally, we performed a study comparing the performance of machines on intoxication versus the performance of human raters. Here, we have shown that, under certain circumstances, machines may even outperform humans (machines: 0.558 UAR, humans: 0.666 UAR) (Ultes et al., 2011b).

In our experiments on user-adaptive dialogue modelling, we were able to provide evidence in a pilot user study that adapting aspects of the dialogue strategy to the user satisfaction may result in an improved system performance (Ultes et al., 2014a,b). This finding was confirmed in a follow-up study with a simulated user (Ultes et al., 2015a). Furthermore, we provided evidence that using the user satisfaction for modelling the reward may be a reasonable approach and be

favourable compared to standard techniques. We have confirmed this in a follow-up study with a POMDP-based dialogue manager showing that our proposed approach for reward modelling further improves the overall system performance. Finally, simulated experiments on modifying the selection of the next system action by predicting the change in coherence also showed the potential of this approach (Ultes et al., 2015a).

6.2 Future Directions

While we have already made a major step towards user-adaptive dialogue, there are still limitations and open questions. The most important limitation of the presented results is that most approaches have only been tested on one domain. While the approaches have been designed to be domain-independent, it may be expected that applying those to a different domain will not result in different behaviour. Still, this has not been shown yet and should therefore be part of future work.

Regarding the recognition of the user satisfaction (or Interaction Quality), only parameters about the interaction have been used. However, linguistic content in both the system and the user utterances are likely to also have an influence on the user satisfaction. While adding this information would result in a loss of the ability to generalise, it may further improve the recognition performance.

For emotion recognition, we have presented a user-dependent approach taking into account the speaker out of a known set of speakers. However, this assumption poses a considerable restriction on the approach as there exist many situations where there is no fixed set of known speakers. Finding approaches which are able to adapt to the speaker without exactly knowing who the speaker is may therefore be regarded as crucial for further improving user-dependent emotion recognition.

For future work on quality-adaptive dialogue, many adaptation techniques have only been evaluated with a simulated user. However, the same adaptation techniques should be tested with real users. While user simulators offer a good means of evaluating dialogues easily, real users usually give new insight by showing unseen behaviour.

While working on quality-adaptive dialogue, it became clear that the effect of adaptive behaviour is much stronger if the capabilities of the SDS allow more complex dialogues. Thus, within the key aspect of *unrestricted user behaviour* (see Chap. 1), increasing the complexity and functionality of spoken dialogue systems poses an important field of research.

While introducing statistical methods into the dialogue management has already resulted in a major improvement in the functionality and performance (e.g., (Lemon and Pietquin, 2012; Young et al., 2013)), there are still a number of open issues. One is the complexity of the dialogue domain. State-of-the-art dialogue systems are still designed to solve a specific task or to communicate in a pre-set domain. Having an open-domain system, however, is much more favorable. While recent approaches to extend the domain of the dialogue dynamically have shown good results (Gačić et al., 2013b, 2014), they are limited as they only extend a known domain by new attributes. Adding completely new domains (both in structure and content) is still an open issue. Here, the user state information may be utilised, e.g., by automatically evaluating the newly created models.

Furthermore, the interaction between the system and the user is still very rigid regarding the turn-taking behaviour. Usually, only complete utterances are produced and processed. To make

this more flexible, the idea of incremental dialogue systems has emerged allowing the production and processing of partial results of the SDS modules (e.g., (Schlangen and Skantze, 2009)). However, there are many timing issues associated to incremental dialogue: when should a partial result be produced as output of the SDS module? When should a result be processed by the next SDS module? Here, information about the user state may help in answering these questions.

With a vision of a personal assistant, pro-active system behaviour poses another hot topic of research. The questions of *if*, *how*, and *when* pro-active system behaviour is desirable need to be addressed (Nothdurft et al., 2015). The answer to these questions highly depends not only on the linguistic content but also on additional user cues. Applying these questions to pro-activity in multi-party interaction further adds a whole new dimension of complexity. For both, user state information may play a key role in making these types of interaction feasible.

A

Questionnaires

The questionnaire has been removed due to copyright limitations.

Fig. A.1: The SASSI-based questionnaire in German used for the pilot experiment on adapting the grounding strategy to user satisfaction.

The questionnaire has been removed due to copyright limitations.

Fig. A.2: The SASSI questionnaire for evaluation of spoken human-machine interaction.

B

Additional Figures and Tables

1. The rater should try to mirror the user's point of view on the interaction as objectively as possible.
2. An exchange consists of the system prompt and the user response. Due to system design, the latter is not always present.
3. The IQ score is defined on a five-point scale with "1=extremely unsatisfied', "2=very unsatisfied", "3=unsatisfied", "4=slightly unsatisfied" and "5=satisfied".
4. The Interaction Quality is to be rated for each exchange in the dialog. The history of the dialog should be kept in mind when assigning the score. For example, a dialog that has proceeded fairly poor for a long time, should require some time to recover.
5. A dialog always starts with an Interaction Quality score of "5".
6. The first user input should also be rated with 5, since until this moment, no rateable interaction has taken place.
7. A request for help does not invariably cause a lower Interaction Quality, but can result in it.
8. In general, the score from one exchange to the following exchange is increased or decreased by one point at the most.
9. Exceptions, where the score can be decreased by two points are, e.g., hot anger or sudden frustration. The rater's perception is decisive here.
10. Also, if the dialog obviously collapses due to system or user behaviour, the score can be set to "1" immediately. An example therefore is a reasonable frustrated sudden hang-up.
11. Anger does not need to influence the score, but can. The rater should try to figure out whether anger was caused by the dialog behaviour or not.
12. In the case a user realises that he should adapt his dialog strategy to obtain the desired result or information and succeeded that way, the Interaction Quality score can be raised up to two points per turn. In other words, the user realizes that he caused the poor Interaction Quality by himself.
13. If the system does not reply with a bus schedule to a specific user query and prompts that the request is out of scope, this can nevertheless be considered as "task completed". Therefore this does not need to affect the Interaction Quality.
14. If a dialog consists of several independent queries, each query's quality is to be rated independently. The former dialog history should not be considered when a new query begins. However, the score provided for the first exchange should be equal to the last label of the previous query.
15. If a constantly low-quality dialog finishes with a reasonable result, the Interaction Quality may be increased.

Fig. B.1: Rater guidelines for annotating Interaction Quality as published in (Schmitt et al., 2012)

Table B.1: All automatically derivable features of the IQ paradigm along with the SDS module they have been derived from (**ASR**, **SLU**, or **Dialog Manager (DM)**) for all three parameter levels (cf. Fig. 4.3) as published in (Schmitt et al., 2012).

	exchange level	
ASR	ASRRECOGNITIONSTATUS	one of 'success', 'reject', 'timeout'
	ASRCONFIDENCE	confidence of the ASR
	BARGED-IN?	did the user barge-in?
	MODALITY	one of 'speech', 'DTMF'
	EXMO	the modality expected from the system ('speech', 'DTMF', 'both')
	UNEXMO?	did the user employ another modality than expected?
	UTTERANCE	raw ASR transcription
	WPUT	number of words per user turn
	UTD	utterance turn duration
SLU	SEMANTICPARSE	semantic interpretation of utterance
	HELPREQUEST?	is the current turn a help request?
DM	ACTIVITY	identifier of the current system action
	ACTIVITYTYPE	one of 'Announcement', 'Question', 'Confirmation', 'wait'
	PROMPT	system prompt
	WPST	number of words per system turn
	REPROMPT?	is the current system turn a reprompt?
	ROLENAME	whether the role of the current system prompt is a confirmation to elicit common ground between user and system
	ROLEINDEX	number of times the role of the current system prompt has remained the same
	LOOPNAME	name of the status of the current system prompt; corresponds with ROLEINDEX
	DD	dialog duration up to this point in seconds
	dialog level	
ASR	MEANASRCONFIDENCE	average of ASR confidence scores
	#ASRSUCCESS	number of exchanges with ASRRECOGNITIONSTATUS 'success'
	%ASRSUCCESS	rate of exchanges with ASRRECOGNITIONSTATUS 'success'
	#ASRREJECTIONS	number of exchanges with ASRRECOGNITIONSTATUS 'reject'
	%ASRREJECTIONS	rate of exchanges with ASRRECOGNITIONSTATUS 'reject'
	#TIME-OUTPROMPTS	number of exchanges with ASRRECOGNITIONSTATUS 'timeout'
	%TIME-OUTPROMPTS	rate of exchanges with ASRRECOGNITIONSTATUS 'timeout'
	#TIME-OUTREJECTIONS	number of exchanges with ASRRECOGNITIONSTATUS 'timeout' or 'reject'
	%TIME-OUTREJECTIONS	rate of exchanges with ASRRECOGNITIONSTATUS 'timeout' or 'reject'
	#BARGE-INS	number of barge-ins
	%BARGE-INS	rate of barge-ins
	#UNEXMO	number of turns with unexpected modality
	%UNEXMO	rate of turns with unexpected modality
SLU	#HELPREQUESTS	number of turns with help request
	%HELPREQUESTS	rate of turns with help request
DM	#REPROMPTS	number of turns being a reprompt
	%REPROMPTS	rate of turns being a reprompt
	#EXCHANGES	number of system-user exchanges
	#SYSTEMTURNS	number of system turns
	#USERTURNS	number of user turns
	#SYSTEMQUESTIONS	number of turns being a system question
	window level	
ASR	{MEAN}ASRCONFIDENCE	average of ASR confidence scores
	{#}ASRSUCCESS	number of successfully parsed user utterances
	{#}ASRREJECTIONS	number of exchanges with ASRRECOGNITIONSTATUS 'reject'
	{#}TIME-OUTPROMPTS	number of exchanges with ASRRECOGNITIONSTATUS 'timeout'
	{#}TIME-OUTREJECTIONS	number of exchanges with ASRRECOGNITIONSTATUS 'timeout' or 'reject'
	{#}BARGE-INS	number of barge-ins
	{#}UNEXMO	number of turns with unexpected modality
SLU	{#}HELPREQUESTS	number of turns where user requested help
DM	{#}REPROMT	number of turns with reprompt
	{#}SYSTEMQUESTIONS	number of turns where ACTIVITYTYPE is 'question'

Table B.2: The three feature sets for IQ recognition and the parameters they contain grouped by the corresponding SDS module and the parameter level.

		BASE*all*	BASE*di*	BASE*25*
	exchange level			
ASR	ASRRecognitionStatus	×	×	×
	ASRConfidence	×	×	×
	Barged-In?	×	×	
	Modality	×	×	
	ExMo	×	×	
	UnExMo?	×	×	
	Utterance	×		
	WPUT	×	×	×
	UTD	×	×	×
SLU	SemanticParse	×		
	HelpRequest?	×	×	
DM	Activity	×		
	ActivityType	×	×	×
	Prompt	×		
	WPST	×	×	×
	RePrompt?	×	×	×
	RoleName	×	×	×
	RoleIndex	×	×	×
	LoopName	×		
	DD	×	×	×
	dialogue level			
ASR	MeanASRConfidence	×	×	×
	#ASRSuccess	×	×	
	%ASRSuccess	×	×	
	#ASRRejections	×	×	×
	%ASRRejections	×	×	×
	#Time-OutPrompts	×	×	
	%Time-OutPrompts	×	×	
	#Time-OutRejections	×	×	×
	%Time-OutRejections	×	×	×
	#Barge-Ins	×	×	
	%Barge-Ins	×	×	
	#UnExMo	×	×	
	%UnExMo	×	×	
SLU	#HelpRequests	×	×	
	%HelpRequests	×	×	
DM	#RePrompts	×	×	×
	%RePrompts	×	×	×
	#Exchanges	×	×	×
	#SystemTurns	×	×	
	#UserTurns	×	×	
	#SystemQuestions	×	×	×
	window level			
ASR	{Mean}ASRConfidence	×	×	×
	{#}ASRSuccess	×	×	×
	{#}ASRRejections	×	×	×
	{#}Time-OutPrompts	×	×	
	{#}Time-OutRejections	×	×	×
	{#}Barge-Ins	×	×	
	{#}UnExMo	×	×	
SLU	{#}HelpRequests	×	×	×
DM	{#}RePrompt	×	×	×
	{#}SystemQuestions	×	×	×

References

(2003) Longman dictionary of contemporary english (4th edition). Pearson Education Ltd.

Allen JF (1995) Natural language understanding (2nd ed.). Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA

Andersen O, Kuhn R, Lazaridès A, Dalsgaard P, Haas J, Nöth E (1996) Comparison of two tree-structured approaches for grapheme-to-phoneme conversion. In: Proceedings of the 4th International Conference on Spoken Language (ICSLP), IEEE, vol. 3, pp. 1700–1703

Antoniou G, van Harmelen F (2004) Web Ontology Language: OWL. In: Staab S, Studer R (eds.) Handbook on Ontologies, Springer Berlin Heidelberg, International Handbooks on Information Systems, pp. 67–92, DOI {10.1007/978-3-540-24750-0_4}

Batliner A, Fischer K, Huber R, Spilker J, Nöth E (2000) Desperately seeking emotions: Actors, wizards, and human beings. In: Cowie R, Douglas-Cowie E, Schröder M (eds.) ISCA Tutorial and Research Workshop (ITRW) on Speech and Emotion, ISCA, ISCA, pp. 195–200

Batliner A, Hacker C, Steidl S, Nöth E, D'Arcy S, Russell M, Wong M (2004) "You stupid tin box" - children interacting with the AIBO robot: A cross-linguistic emotional speech corpus. In: Proceedings of the 4th International Conference of Language Resources and Evaluation LREC 2004, ELRA, pp. 171–174, URL http://www5.informatik.uni-erlangen.de/Forschung/Publikationen/2004/Batliner04-YST.pdf

Beyerer J (2008) University course pattern classification

Bickmore TW, Puskar K, Schlenk EA, Pfeifer LM, Sereika SM (2010) Maintaining reality: Relational agents for antipsychotic medication adherence. Interacting with Computers 22:276–288

Bocklet T, Riedhammer K, Nöth E (2011) Drink and Speak: On the Automatic Classification of Alcohol Intoxication by Acoustic, Prosodic and Text-Based Features. In: Proceedings of the 12th Annual Conference of the International Speech Communication Association (INTERSPEECH), ISCA, pp. 3213–3216

Boersma P (2002) Praat, a system for doing phonetics by computer. Glot international 5(9/10):341–345

Bohlin P, Cooper R, Engdahl E, Larsson S (1999) Information states and dialogue move engines. In: IJCAI Workshop on knowledge and reasoning in practical dialogue systems, AAAI

Bohus D, Rudnicky AI (2003) RavenClaw: Dialog Management Using Hierarchical Task Decomposition and an Expectation Agenda. In: Proceedings of the 8th European Conference on Speech Communication and Technology

Bone D, Black M, Li M, Metallinou A, Lee S, Narayanan SS (2011) Intoxicated speech detection by fusion of speaker normalized hierarchical features and gmm supervectors. In: Proceedings of the 12th Annual Conference of the International Speech Communication Association (INTERSPEECH), ISCA, pp. 3217–3220

Boularias A, Chinaei HR, Chaib-draa B (2010) Learning the Reward Model of Dialogue POMDPs from Data. In: NIPS Workshop on Machine Learning for Assistive Techniques

Bui TH, Poel M, Nijholt A, Zwiers J (2007) A POMDP approach to Affective Dialogue Modeling. In: Proceedings of the Workshop on Fundamentals of Verbal and Non-Verbal Communication and the Biometric Issue, Ios Press Inc, vol. 18, pp. 349–355

Bui TH, Poel M, Nijholt A, Zwiers J (2009) A tractable hybrid DDN-POMDP approach to affective dialogue modeling for probabilistic frame-based dialogue systems. Natural Language Engineering 15(02):273–307

Burkhardt F, Rolfes M, Sendlmeier W, Weiss B (2005) A database of german emotional speech. In: Proceedings of the International Conference on Speech and Language Processing (Interspeech), ISCA, pp. 1517–1520, URL http://felix.syntheticspeech.de/publications/databaseOfGermanEmotionalSpeech.pdf

Carpenter B (1992) The logic of typed feature structures. Cambridge University Press, New York, NY, USA

Chai J, Horvath V, Nicolov N, Stys M, Kambhatla N, Zadrozny W, Melville P (2002) Natural Language Assistant: A Dialog System for Online Product Recommendation. AI Magazine 23:63–75

Chang CC, Lin CJ (2011) LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2:27:1–27:27, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: Synthetic Minority Over-sampling TEchnique. Journal of Artificial Intelligence Research 16:321–357

Chu-Carroll J (2000) MIMIC: An adaptive mixed initiative spoken dialogue system for information queries. In: Proceedings of the sixth conference on Applied natural language processing, Association for Computational Linguistics, pp. 97–104

Cohen J (1960) A coefficient of agreement for nominal scales. In: Educational and Psychological Measurement, vol. 20, pp. 37–46

Cohen J (1968) Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. Psychological bulletin 70(4):213

Cohen WW (1995) Fast effective rule induction. In: Proceedings of the 12th International Conference on Machine Learning, Morgan Kaufmann, pp. 115–123

Conati C, Maclaren H (2005) Data-Driven Refinement of a Probabilistic Model of User Affect. In: Ardissono L, Brna P, Mitrovic A (eds.) User modeling 2005, Lecture Notes in Computer Science, vol. 3538, Springer Berlin Heidelberg, pp. 40–49

Cornelius R (1996) The science of emotion: research and tradition in the psychology of emotions. Prentice Hall, Upper Saddle River, N.J., London

Crockford D (2006) RFC 4627 - The application/json Media Type for JavaScript Object Notation (JSON). Tech. rep., IETF, URL http://tools.ietf.org/html/rfc4627

D'Mello S, Olney A, Williams C, Hays P (2012) Gaze tutor: A gaze-reactive intelligent tutoring system. International Journal of Human-Computer Studies 70(5):377–398

Doll WJ, Torkzadeh G (1991) The measurement of end-user computing satisfaction: theoretical and methodological issues. MIS quarterly 15:5–10, DOI 10.2307/249429, URL http://portal.acm.org/citation.cfm?id=110110.110114

Duda RO, Hart PE, Stork DG (2001) Pattern Classification, 2nd edn. Wiley-Interscience

El Asri L, Laroche R, Pietquin O (2012) Reward Function Learning for Dialogue Management. In: Proceedings of the 6t Starting AI Researchers' Symposium (STAIRS), IOS Press, pp. 95–106

El Asri L, Laroche R, Pietquin O (2013) Reward shaping for statistical optimisation of dialogue management. In: Statistical Language and Speech Processing, Springer, pp. 93–101

El Asri L, Khouzaimi H, Laroche R, Pietquin O (2014) Ordinal regression for interaction quality prediction. In: International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 3245–3249

Enberg IS, Hansen AV (1996) Documentation of the danish emotional speech database. Tech. rep., Aalborg University, Denmark

Engel Y (2005) Algorithms and representations for reinforcement learning. PhD thesis, Hebrew University

Engel Y, Mannor S, Meir R (2005) Reinforcement learning with gaussian processes. In: Proceedings of the 22nd international conference on Machine learning, ACM, pp. 201–208

Engelbrecht KP (2012) Estimating spoken dialog system quality with user models. Springer Science & Business Media

Engelbrecht KP, Möller S (2010) A User Model to Predict User Satisfaction with Spoken Dialog Systems. In: Lee GG, Mariani J, Minker W, Nakamura S (eds.) Spoken Dialogue Systems for Ambient Environments. 2nd Int. Workshop on Spoken Dialogue Systems Technology, Springer, Lecture Notes in Artificial Intelligence, pp. 150–155

Engelbrecht KP, Gödde F, Hartard F, Ketabdar H, Möller S (2009) Modeling user satisfaction with Hidden Markov Model. In: Proceedings of the 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL), Association for Computational Linguistics, Morristown, NJ, USA, pp. 170–177

Evanini K, Hunter P, Liscombe J, Suendermann D, Dayanidhi K, Pieraccini R (2008) Caller experience: A method for evaluating dialog systems and its automatic prediction. In: Spoken Language Technology Workshop (SLT), IEEE, pp. 129–132

Forbes-Riley K, Litman DJ (2011) Benefits and challenges of real-time uncertainty detection and adaptation in a spoken dialogue computer tutor. Speech Communication 53(9):1115–1136

Forbes-Riley K, Litman DJ (2012) Adapting to Multiple Affective States in Spoken Dialogue. In: Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDial), Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 217–226, URL http://dl.acm.org/citation.cfm?id=2392800.2392839

Francois JM (2006) Jahmm - An implementation of HMM in Java. URL http://code.google.com/p/jahmm/

Fraser NM (1994) The sundial speech understanding and dialogue project: results and implications for translation. In: Aslib proceedings, MCB UP Ltd, vol. 46, pp. 141–148

Gačić M, Young SJ (2014) Gaussian processes for POMDP-based dialogue manager optimization. IEEE/ACM Transactions on Audio, Speech, and Language Processing 22(1):28–40

Gačić M, Jurčíček F, Keizer S, Mairesse F, Thomson B, Yu K, Young SJ (2010) Gaussian processes for fast policy optimisation of POMDP-based dialogue managers. In: Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDial), Association for Computational Linguistics, pp. 201–204

Gačić M, Breslin C, Henderson M, Kim D, Szummer M, Thomson B, Tsiakoulis P, Young SJ (2013a) On-line policy optimisation of Bayesian spoken dialogue systems via human interaction. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 8367–8371

Gačić M, Breslin C, Henderson M, Kim D, Szummer M, Thomson B, Tsiakoulis P, Young SJ (2013b) POMDP-based dialogue manager adaptation to extended domains. In: Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL), Association for Computational Linguistics, Metz, France, pp. 214–222, URL http://www.aclweb.org/anthology/W/W13/W13-4035

Gačić M, Kim D, Tsiakoulis P, Breslin C, Henderson M, Szummer M, Thomson B, Young SJ (2014) Incremental online adaptation of POMDP-based dialogue managers to extended domains. In: Proceedings of the 15th International Conference on Spoken Language Processing (INTERSPEECH), ISCA, pp. 140–144

Gandhe S, Traum D (2008) An evaluation understudy for dialogue coherence models. In: Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue, Association for Computational Linguistics, pp. 172–181

Ginzburg J (1996) Dynamics and the semantics of dialogue. Seligman, Jerry, & Westerståhl, Dag (eds), Logic, language and computation 1

Glodek M, Scherer S, Schwenker F (2011) Conditioned Hidden Markov Model Fusion for Multimodal Classification. In: Proceedings of the 12th Annual Conference of the International Speech Communication Association (INTERSPEECH 2011), International Speech Communication Association, pp. 2269–2272

Gnjatović M, Rösner D (2008) Adaptive Dialogue Management in the NIMITEK Prototype System. In: Proceedings of the 4th tutorial and research workshop on Perception and Interactive Technologies for Speech-Based Systems (PIT), Springer-Verlag, Berlin, Heidelberg, pp. 14–25, DOI 10.1007/978-3-540-69369-7_3

Grimm M, Kroschel K, Mower E, Narayanan SS (2007) Primitives-based evaluation and estimation of emotions in speech. Speech Communication 49(10):787–800

Grimm M, Kroschel K, Narayanan SS (2008) The Vera am Mittag German audio-visual emotional speech database. In: Proceedings of the International Conference on Multimedia and Expo, IEEE, pp. 865–868

Griol D, Hurtado LF, Segarra E, Sanchis E (2008) A statistical approach to spoken dialog systems design and evaluation. Speech Communication 50(8):666–682

Haq S, Jackson PJB (2010) Machine Audition: Principles, Algorithms and Systems, IGI Global, Hershey PA, chap. Multimodal Emotion Recognition, pp. 398–423

Hara S, Kitaoka N, Takeda K (2010) Estimation method of user satisfaction using n-gram-based dialog history model for spoken dialog system. In: Calzolari N, Choukri K, Maegaard B, Mariani J, Odijk J, Piperidis S, Rosner M, Tapias D (eds.) Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10), European Language Resources Association (ELRA), Valletta, Malta

Hassenzahl M, Burmester M, Koller F (2003) AttrakDiff: Ein Fragebogen zur Messung wahrgenommener hedonischer und pragmatischer Qualität. In: Mensch & Computer 2003, Springer, pp. 187–196

He J, Chaparro A, Nguyen B, Burge R, Crandall J, Chaparro B, Ni R, Cao S (2013) Texting while driving: is speech-based texting less risky than handheld texting? In: Proc. of 5th International Conference on Automotive User Interfaces and Interactive Vehicular Applications (Automotive'UI 13), Eindhoven, The Netherlands, pp. 124–130

Heinroth T, Denich D (2011) Spoken interaction within the computed world: Evaluation of a multitasking adaption spoken dialogue system. In: 35th Annual IEEE Computer Software and Applications Conference (COMPSAC), IEEE, pp. 134–143

Heinroth T, Denich D, Schmitt A (2010a) OwlSpeak - Adaptive Spoken Dialogue within Intelligent Environments. In: 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 666–671, DOI 10.1109/PERCOMW.2010.5470518, presented as part of SmartE Workshop

Heinroth T, Denich D, Schmitt A, Minker W (2010b) Efficient spoken dialogue domain representation and interpretation. In: Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC), European Language Resources Association (ELRA), Valetta, Malta, pp. 2445–2449, URL http://www.lrec-conf.org/proceedings/lrec2010/summaries/345.html

Higashinaka R, Minami Y, Dohsaka K, Meguro T (2010a) Issues in predicting user satisfaction transitions in dialogues: Individual differences, evaluation criteria, and prediction models. In: Lee G, Mariani J, Minker W, Nakamura S (eds.) Spoken Dialogue Systems for Ambient Environments, Lecture Notes in Computer Science, vol. 6392, Springer Berlin / Heidelberg, pp. 48–60, 10.1007/978-3-642-16202-2_5

Higashinaka R, Minami Y, Dohsaka K, Meguro T (2010b) Modeling user satisfaction transitions in dialogues from overall ratings. In: Proceedings of the SIGDIAL 2010 Conference, Association for Computational Linguistics, Tokyo, Japan, pp. 18–27

Hofmann H, Silberstein A, Ehrlich U, Berton A, Muller C, Mahr A (2014) Development of Speech-Based In-Car HMI Concepts for Information Exchange Internet Apps. In: Natural Interaction with Robots, Knowbots and Smartphones: Putting Spoken Dialog Systems into Practice, Springer, pp. 15–28

Hone KS, Graham R (2000) Towards a tool for the Subjective Assessment of Speech System Interfaces (SASSI). Natural Language Engineering 6(3-4):287–303, DOI 10.1017/s1351324900002497

Hönig F, Batliner A, Nöth E (2011) Does it Groove or does it Stumble-Automatic Classification of Alcoholic Intoxication using Prosodic Features. In: Proceedings of the 12th Annual Conference of the International Speech Communication Association (INTERSPEECH), ISCA, pp. 3225–3228

Horchak OV, Giger JC, Cabral M, Pochwatko G (2014) From demonstration to theory in embodied language comprehension: A review. Cognitive Systems Research 29-30:66–85

ISO (1998) Ergonomic requirements for office work with visual display terminals (VDTs), Part 11: Guidance on usability. International Standardization Organization (ISO)

ITU (1994) Terms and definitions related to quality of service and network performance including dependability. ITU-T Recommendation E.800, International Telecommunication Union, Geneva, Switzerland

ITU (2003) Subjective quality evaluation of telephone services based on spoken dialogue systems. ITU-T Recommendation P.851, International Telecommunication Union, Geneva, Switzerland

ITU (2007) Vocabulary for performance and quality of service. ITU-T Amendment 1 to P.10/G.100, International Telecommunication Union, Geneva, Switzerland

Ives B, Olson MH, Baroudi JJ (1983) The measurement of user information satisfaction. Communications of the ACM 26:785–793, DOI 10.1145/358413.358430

Jokinen K, Kanto K (2004) User expertise modelling and adaptivity in a speech-based e-mail system. In: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics, URL http://anthology.aclweb.org/P/P04/P04-1012.pdf

Jurafsky D, Martin JH (2008) Speech and Language Processing, 2nd edn. Prentice Hall

Kaelbling LP, Littman ML, Cassandra AR (1998) Planning and acting in partially observable stochastic domains. Artificial Intelligence 101(1-2):99–134

Klein J, Youngme M, Picard RW (2002) This computer responds to user frustration: Theory, design, and results. Interacting with Computers 14:119–140

Kopp KJ, Britt MA, Millis K, Graesser AC (2012) Improving the efficiency of dialogue in tutoring. Learning and Instruction 22(5):320–330

Lamel L, Minker W, Paroubek P (2000) Towards best practice in the development and evaluation of speech recognition components of a spoken language dialog system. Natural Language Engineering 6:305–322, DOI 10.1017/s1351324900002515, URL http://portal.acm.org/citation.cfm?id=973935.973942

Larcker D, Lessig VP (1980) Perceived usefulness of information: A psychometric examination. Decision Sciences pp. 121–134

Larsson S, Traum DR (2000) Information state and dialogue management in the TRINDI Dialogue Move Engine. Natural Language Engineering Special Issue 6:323–340, URL http://www.ling.gu.se/~sl/nle.ps

Lee CM, Narayanan SS (2005) Toward detecting emotions in spoken dialogs. IEEE Transactions on Speech and Audio Processing 13(2):293–303, DOI 10.1109/tsa.2004.838534

Lee CM, Narayanan SS, Pieraccini R (2001) Recognition of negative emotions from the speech signal. In: IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), pp. 240–243

Lee CM, Narayanan SS, Pieraccini R (2002) Combining acoustic and language information for emotion. Proceedings of the 7th International Conference on Spoken Language Processing 2002

Lee S, Eskenazi M (2012) An unsupervised approach to user simulation: toward self-improving dialog systems. In: Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue, Association for Computational Linguistics, pp. 50–59

Lefevre F, Gačić M, Jurčíček F, Keizer S, Mairesse F, Thomson B, Yu K, Young SJ (2009) k-nearest neighbor Monte-Carlo control algorithm for POMDP-based dialogue systems. In: Proceedings of the SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue, Association for Computational Linguistics, pp. 272–275

Leggetter CJ, Woodland PC (1995) Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. Computer Speech & Language 9(2):171–185, DOI 10.1006/csla.1995.0010, URL http://www.sciencedirect.com/science/article/pii/S0885230885700101

Lemon O, Pietquin O (2012) Data-Driven Methods for Adaptive Spoken Dialogue Systems. Springer New York, DOI 10.1007/978-1-4614-4803-7

Lindgaard G, Dudek C (2003) What is this evasive beast we call user satisfaction? Interacting with Computers 15(3):429–452

Litman DJ, Forbes-Riley K (2014) Evaluating a spoken dialogue system that detects and adapts to user affective states. In: 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue, pp. 181–185

Litman DJ, Pan S (2002) Designing and evaluating an adaptive spoken dialogue system. User Modeling and User-Adapted Interaction 12(2-3):111–137, DOI 10.1023/a:1015036910358

Litman DJ, Silliman S (2004) ITSPOKE: An Intelligent Tutoring Spoken Dialogue System. In: Proc. of Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (HLT/NAACL), Boston, USA, pp. 233–236

Mann HB, Whitney DR (1947) On a test of whether one of two random variables is stochastically larger than the other. The annals of mathematical statistics 18(1):50–60

Matheson C, Poesio M, Traum D (2000) Modelling grounding and discourse obligations using update rules. In: Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference, Association for Computational Linguistics, pp. 1–8

McTear MF (2004) Spoken Dialogue Technology: Towards the Conversational User Interface. Springer, London

Meguro T, Higashinaka R, Minami Y, Dohsaka K (2010) Controlling listening-oriented dialogue using partially observable Markov decision processes. In: Proceedings of the 23rd International Conference on Computational Linguistics, Association for Computational Linguistics, pp. 761–769

Mehrabian A (1996) Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in temperament. Current Psychology 14(4):261–292, DOI 10.1007/bf02686918

Metze F, Ajmera J, Englert R, Bub U, Burkhardt F, Stegmann J, Müller C, Huber R, Andrassy B, Bauer J, Littel B (2007) Comparison of four approaches to age and gender recognition. In: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 1, pp. 1089–1092

Metze F, Anguera X, Barnard E, Davel M, Gravier G (2014) Language independent search in MediaEval's Spoken Web Search task. Computer Speech and Language 28(5):1066–1082

Miller S, Bobrow R, Ingria R, Schwartz R (1994) Hidden understanding models of natural language. In: Proceedings of the 32nd annual meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Morristown, NJ, USA, pp. 25–32, DOI 10.3115/981732.981736

Minker W, Waibel A, Mariani J (1999) Stochastically-Based Semantic Analysis. Kluwer Academic Publishers, Norwell, MA, USA

Minker W, López-Cózar R, McTear MF (2009) The role of spoken language dialogue interaction in intelligent environments. Journal of Ambient Intelligence and Smart Environments 1(1):31–36

Minker W, Heinroth T, Strauss PM, Zaykovskiy D (2010) Spoken dialogue systems for intelligent environments. In: Human-Centric Interfaces for Ambient Intelligence, Elsevier, pp. 453–478

Möller S (2005) Quality of Telephone-based Spoken Dialogue Systems. Springer, New York

Möller S, Engelbrecht KP, Schleicher R (2008) Predicting the quality and usability of spoken dialogue services. Speech Communication 50(8-9):730–744, DOI 10.1016/j.specom.2008.03.001

Möller S, Engelbrecht KP, Kühnel C, Wechsung I, Weiss B (2009) A taxonomy of Quality of Service and Quality of Experience of multimodal human-machine interaction. In: International Workshop on Quality of Multimedia Experience (QoMEx), pp. 7–12, DOI 10.1109/qomex.2009.5246986

Monahan GE (1982) State of the art—a survey of partially observable Markov decision processes: theory, models, and algorithms. Management Science 28(1):1–16

Montacié C, Caraty MJ (2011) Combining multiple phoneme-based classifiers with audio feature-based classifier for the detection of alcohol intoxication. In: Proceedings of the 12th Annual Conference of the International Speech Communication Association

Mori H, Satake T, Nakamura M, Kasuya H (2011) Constructing a spoken dialogue corpus for studying paralinguistic information in expressive conversation and analyzing its statistical/acoustic characteristics. Speech Communication 53(1):36–50

Muir BM (1994) Trust in automation: Part i. theoretical issues in the study of trust and human intervention in automated systems. Ergonomics 37(11):1905–1922

Müller C, Wasinger R (2002) Adapting Multimodal Dialog for the Elderly. In: ABIS Workshop on Personalization for the Mobile World, Hannover, Germany, pp. 31–34

Nogueiras Rodríguez A (2011) An HMM-based approach to the INTERSPEECH 2011 speaker state challenge. In: Proceedings of the 12th Annual Conference of the International Speech Communication Association (INTERSPEECH), pp. 3289–3292

Noh H, Lee S, Kim K, Lee K, Lee GG (2011) Ranking dialog acts using discourse coherence indicator for language tutoring dialog systems. In: Proceedings of the Paralinguistic Information and its Integration in Spoken Dialogue Systems Workshop, Springer, pp. 203–214

Nothdurft F, Honold F, Kurzok P (2012) Using explanations for runtime dialogue adaptation. In: Proceedings of the 14th ACM International Conference on Multimodal Interaction, ACM, pp. 63–64

Nothdurft F, Ultes S, Minker W (2015) Finding appropriate interaction strategies for proactive dialogue systems—an open quest. In: Proceedings of the 2nd European and the 5th Nordic Symposium on Multimodal Communication, August 6-8, 2014, Tartu, Estonia, LiU Electronic Press, 110, pp. 73–80

Nothdurft F, Ultes S, Minker W (2016) User-centred spoken dialogue management. In: Next Generation Intelligent Environments, Springer, pp. 265–294

Oshry M, Auburn R, Baggia P, Bodell M, Burke D, Burnett D, Candell E, Carter J, Mcglashan S, Lee A, Porter B, Rehor K (2007) Voice Extensible Markup Language (VoiceXML) Version 2.1. Tech. rep., W3C - Voice Browser Working Group

Paek T, Pieraccini R (2008) Automating spoken dialogue management design using machine learning: An industry perspective. Speech communication 50(8):716–729

Perrault CR, Allen JF, Cohen PR (1978) Speech acts as a basis for understanding dialogue coherence. In: Proceedings of the 1978 workshop on Theoretical issues in natural language processing, Association for Computational Linguistics, pp. 125–132

Petrushin VA (1999) Emotion in speech: Recognition and application to call centers. In: Artificial Neural Networks in Engineering (ANNIE '99), St. Louis, pp. 7–10

Pittermann J, Pittermann A, Minker W (2009) Handling Emotions in Human-Computer Dialogues. Springer, Dordrecht (The Netherlands), URL http://www.springer.com/linguistics/computational+linguistics/book/978-90-481-3128-0

Platt JC (1999) Fast training of support vector machines using sequential minimal optimization, MIT Press, Cambridge, MA, USA, pp. 185–208. URL http://portal.acm.org/citation.cfm?id=299094.299105

Plutchik R (1980) Emotion: A Psychoevolutionary Synthesis. Harper & Row

Polzehl T, Schmitt A, Metze F, Wagner M (2011) Anger recognition in speech using acoustic and linguistic cues. Speech Communication 53(9-10):1198–1209, DOI 10.1016/j.specom.2011.05.002, URL http://www.sciencedirect.com/science/article/pii/S0167639311000677

Potel M (1996) MVP: Model-View-Presenter The Taligent Programming Model for C++ and Java. Tech. rep., Taligent Inc, URL http://www.wildcrest.com/Potel/Portfolio/mvp.pdf

Prakken H (2005) Coherence and flexibility in dialogue games for argumentation. Journal of logic and computation 15(6):1009–1040

Purandare A, Litman DJ (2008) Analyzing dialog coherence using transition patterns in lexical and semantic features. In: FLAIRS Conference, pp. 195–200

Qu C, Brinkman WP, Ling Y, Wiggers P, Heynderickx I (2014) Conversations with a virtual human: Synthetic emotions and human responses. Computers in Human Behavior 34:58–68

Quinlan JR (1992) C4.5: Programs for Machine Learning, 1st edn. Morgan Kaufmann Series in Machine Learning, Morgan Kaufmann, URL http://www.worldcat.org/isbn/1558602380

Rabiner LR (1989) A tutorial on hidden Markov models and selected applications in speech recognition. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA

Raux A, Langner B, Black AW, Eskenazi M (2003) LETS GO: Improving spoken dialog systems for the elderly and non-native,. In: Proceedings of Eurospeech 2003, Citeseer

Raux A, Langner B, Bohus D, Black AW, Eskenazi M (2005) Let's go public! taking a spoken dialog system to the real world. In: Proceedings of Interspeech 2005

Raux A, Bohus D, Langner B, Black AW, Eskenazi M (2006) Doing research on a deployed spoken dialogue system: One year of let's go! experience. In: Proc. of the International Conference on Speech and Language Processing (ICSLP)

Reeves B, Nass C (1996) The media equation: how people treat computers, television, and new media like real people and places. Cambridge University Press, New York, NY, USA

Rieser V, Lemon O (2008a) Automatic learning and evaluation of user-centered objective functions for dialogue system optimisation. In: Calzolari N, Choukri K, Maegaard B (eds.) Proceedings of the Sixth International Language Resources and Evaluation (LREC'08), European Language Resources Association (ELRA), Marrakech, Morocco, pp. 2356–2361, http://www.lrec-conf.org/proceedings/lrec2008/

Rieser V, Lemon O (2008b) Learning effective multimodal dialogue strategies from wizard-of-oz data: Bootstrapping and evaluation. In: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, ACL, pp. 638–646

Rosenblatt F (1958) The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review 65(6):386–408

Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. Nature 323(6088):533–536, DOI 10.1038/323533a0

Russell S (1998) Learning agents for uncertain environments. In: Proceedings of the eleventh annual conference on Computational learning theory, ACM, pp. 101–103

Saz O, Yin SC, Lleida E, Rose R, Vaquero C, Rodríguez WR (2009) Tools and Technologies for Computer-Aided Speech and Language Therapy. Speech Communication 51(10):948–967

Schatzmann J, Young SJ (2009) The hidden agenda user simulation model. Audio, Speech, and Language Processing, IEEE Transactions on 17(4):733–747

Scherer K (2002) Emotion. In: Sozialpsychologie, Springer, pp. 165–213

Schiel F (2011) Perception of alcoholic intoxication in speech. In: Proceedings of the Interspeech, pp. 3281–3284

Schiel F, Heinrich C, Barfüßer S, Gilg T (2008) ALC: Alcohol Language Corpus. In: Calzolari N, Choukri K, Maegaard B (eds.) Proceedings of the Sixth International Language Resources and Evaluation (LREC'08), European Language Resources Association (ELRA), Marrakech, Morocco, pp. 1641–1645

Schlangen D, Skantze G (2009) A general, abstract model of incremental dialogue processing. In: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, Stroudsburg, PA, USA, EACL '09, pp. 710–718, URL http://dl.acm.org/citation.cfm?id=1609067.1609146

Schmitt A, Minker W (2013) Towards Adaptive Spoken Dialog Systems. Springer-Verlag New York, DOI 10.1007/978-1-4614-4593-7, URL http://www.springer.com/engineering/signals/book/978-1-4614-4592-0

Schmitt A, Ultes S (2015) Interaction quality: Assessing the quality of ongoing spoken dialog interaction by experts—and how it relates to user satisfaction. Speech Communication 74:12–36, DOI 10.1016/j.specom.2015.06.003, URL http://www.sciencedirect.com/science/article/pii/S0167639315000679

Schmitt A, Heinroth T, Bertrand G (2009a) Towards Emotion, Age- and Gender-Aware VoiceXML Applications. In: 5th International Conference on Intelligent Environments (IE'09), pp. 34–41

Schmitt A, Heinroth T, Liscombe J (2009b) On nomatchs, noinputs and bargeins: Do non-acoustic features support anger detection? In: Proceedings of the 10th Annual SIGDIAL Meeting on Discourse and Dialogue, Association for Computational Linguistics, London (UK), pp. 128–131

Schmitt A, Pieraccini R, Polzehl T (2010a) Advances in Speech Recognition: Mobile Environments, Call Centers and Clinics, Springer US, chap. 'For Heaven's sake, gimme a live person!' Designing Emotion-Detection Customer Care Voice Applications in Automated Call Cent, pp. 191–219. DOI 10.1007/978-1-4419-5951-5_9

Schmitt A, Tschaffon U, Heinroth T, Minker W (2010b) Inter-labeler agreement for anger detection in interactive voice response systems. In: Proceedings of the 6th International Conference on Intelligent Environments (IE), IEEE, pp. 112–115, DOI 10.1109/IE.2010.28

Schmitt A, Schatz B, Minker W (2011a) Modeling and predicting quality in spoken human-computer interaction. In: Proceedings of the SIGDIAL 2011 Conference, Association for Computational Linguistics, Portland, Oregon, USA, pp. 173–184

Schmitt A, Schatz B, Minker W (2011b) A statistical approach for estimating user satisfaction in spoken human-machine interaction. In: Proceedings of the IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), IEEE, Amman, Jordan, pp. 1–6

Schmitt A, Ultes S, Minker W (2012) A parameterized and annotated spoken dialog corpus of the cmu let's go bus information system. In: International Conference on Language Resources and Evaluation (LREC), pp. 3369–337

Schuller B (2006) Automatische emotionserkennung aus sprachlicher und manueller interaktion. Dissertation, Technische Universität München, München

Schuller B, Steidl S, Batliner A (2009a) The interspeech 2009 emotion challenge. In: Proc. of the International Conference on Speech and Language Processing (ICSLP)

Schuller B, Vlasenko B, Eyben F, Rigoll G, Wendemuth A (2009b) Acoustic emotion recognition: A benchmark comparison of performances. In: Workshop on Automatic Speech Recognition & Understanding (ASRU), IEEE, pp. 552–557

Schuller B, Steidl S, Batliner A, Schiel F, Krajewski J (2011) The interspeech 2011 speaker state challenge. In: Proc. of the 12th Annual Conference of the International Speech Communication Association (INTERSPEECH 2011), pp. 3201–3204

Seneff S, Polifroni J (2000) Dialogue management in the mercury flight reservation system. In: Proceedings of the 2000 ANLP/NAACL Workshop on Conversational systems-Volume 3, Association for Computational Linguistics, pp. 11–16

Shafran I, Riley M, Mohri M (2003) Voice signatures. In: IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), pp. 31–36, DOI 10.1109/asru.2003.1318399

Sidorov M, Ultes S, Schmitt A (2014a) Comparison of gender- and speaker-adaptive emotion recognition. In: International Conference on Language Resources and Evaluation (LREC), pp. 3476–3480

Sidorov M, Ultes S, Schmitt A (2014b) Emotions are a personal thing: Towards speaker-adaptive emotion recognition. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pp. 4836–4840

Skantze G, Hjalmarsson A, Oertel C (2014) Turn-taking, feedback and joint attention in situated human-robot interaction. Speech Communication 65:50–66

Spearman CE (1904) The proof and measurement of association between two things. American Journal of Psychology 15:88–103

Stent A, Stenchikova S, Marge M (2006) Reinforcement learning of dialogue strategies with hierarchical abstract machines. In: Proc. of Spoken Language Technology Workshop (SLT'06), Palm Beach, Aruba, pp. 210–213

Thomson B, Young SJ (2010) Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. Computer Speech & Language 24(4):562–588

Thomson B, Yu K, Keizer S, Gačić M, Jurčíček F, Mairesse F, Young SJ (2010) Bayesian Dialogue System for the Let's Go Spoken Dialogue Challenge. In: Proceedings of the IEEE Workshop on Spoken Language Technology, URL http://mi.eng.cam.ac.uk/~sjy/papers/tykg10.pdf

Tsai MJ (2005) The VoiceXML dialog system for the e-commerce ordering service. In: Proc. of 9th International Conference on Computer Supported Cooperative Work in Design (CSCWD'05), Coventry, UK, pp. 95–100

Ultes S, Minker W (2013a) HIS-OwlSpeak: A Model-driven Dialogue Manager With Multiple Control Modes. In: Proceedings of the 9th International Conference on Intelligent Environments (IE), IEEE, pp. 108–115

Ultes S, Minker W (2013b) Improving interaction quality recognition using error correction. In: Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue, Association for Computational Linguistics, pp. 122–126, URL http://www.aclweb.org/anthology/W/W13/W13-4018

Ultes S, Minker W (2013c) Interaction quality: A review. Bulletin of Siberian State Aerospace University named after academician MF Reshetnev (4):153–156, URL http://www.vestnik.sibsau.ru/images/vestnik/ves450.pdf

Ultes S, Minker W (2014a) Interaction Quality Estimation in Spoken Dialogue Systems Using Hybrid-HMMs. In: Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL), Association for Computational Linguistics, pp. 208–217, URL http://www.aclweb.org/anthology/W14-4328

Ultes S, Minker W (2014b) Managing adaptive spoken dialogue for intelligent environments. Journal of Ambient Intelligence and Smart Environments 6(5):523–539, DOI 10.3233/ais-140275

Ultes S, Heinroth T, Schmitt A, Minker W (2011a) A theoretical framework for a user-centered spoken dialog manager. In: Proceedings of the Paralinguistic Information and its Integration in Spoken Dialogue Systems Workshop, Springer, pp. 241 – 246

Ultes S, Schmitt A, Minker W (2011b) Attention, Sobriety Checkpoint! Can Humans Determine by Means of Voice, if Someone is Drunk... and can Automatic Classifiers Compete? In: Proceedings of the 12th Annual Conference of the International Speech Communication Association (INTERSPEECH), ISCA, pp. 3221–3224

Ultes S, ElChabb R, Minker W (2012a) Application and Evaluation of a Conditioned Hidden Markov Model for Estimating Interaction Quality of Spoken Dialogue Systems. In: Mariani J, Devillers L, Garnier-Rizet M, Rosset S (eds.) Proceedings of the 4th International Workshop on Spoken Language Dialog System (IWSDS), Springer, pp. 141–150

Ultes S, Schmitt A, ElChabb R, Minker W (2012b) Statistical modeling of interaction quality in spoken dialogue systems: A comparison of (conditioned) hidden-markov-model-based classifiers vs. support vector machines. Bulletin of Siberian State Aerospace University named after academician MF Reshetnev pp. 115–118

Ultes S, Schmitt A, Minker W (2012c) Towards quality-adaptive spoken dialogue management. In: NAACL-HLT Workshop on Future directions and needs in the Spoken Dialog Community: Tools and Data (SDCTD 2012), Association for Computational Linguistics, Montréal, Canada, pp. 49–52, URL http://www.aclweb.org/anthology/W12-1819

Ultes S, ElChabb R, Schmitt A, Minker W (2013a) JaCHMM: A Java-Based Conditioned Hidden Markov Model Library. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 3213–3217

Ultes S, Schmitt A, Minker W (2013b) On quality ratings for spoken dialogue systems – experts vs. users. In: Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, pp. 569–578

Ultes S, Dikme H, Minker W (2014a) Dialogue management for user-centered adaptive dialogue. In: Proceedings of the 5th International Workshop On Spoken Dialogue Systems (IWSDS), URL http://www.uni-ulm.de/fileadmin/website_uni_ulm/allgemein/2014_iwsds/iwsds2014_lp_ultes.pdf

Ultes S, Dikme H, Minker W (2014b) First insight into quality-adaptive dialogue. In: International Conference on Language Resources and Evaluation (LREC), pp. 246–251

Ultes S, Kraus M, Schmitt A, Minker W (2015a) Quality-adaptive spoken dialogue initiative selection and implications on reward modelling. In: Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL), ACL, pp. 374–383

Ultes S, Platero Sánchez MJ, Schmitt A, Minker W (2015b) Analysis of an Extended Interaction Quality Corpus. In: Lee GG, Kim HK, Jeong M, Kim JH (eds.) Natural Language Dialog Systems and Intelligent Assistants, Springer International Publishing, pp. 41–52, DOI 10.1007/978-3-319-19291-8_4, URL https://www.uni-ulm.de/fileadmin/website_uni_ulm/allgemein/2015_iwsds/iwsds2015_submission_4.pdf

Vapnik VN (1995) The nature of statistical learning theory. Springer-Verlag New York, Inc., New York, NY, USA

Viterbi A (1967) Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Transactions on Information Theory 13(2):260–269, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1054010

Vogt T, André E (2006) Improving automatic emotion recognition from speech via gender differentiation. In: Proceedings of 5th International Conference on Language Resources and Evaluation (LREC), pp. 1123–1126

Walker M (2000) An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. Journal of Artificial Intelligence Research 12:387–416

Walker M, Litman DJ, Kamm CA, Abella A (1997) PARADISE: a framework for evaluating spoken dialogue agents. In: Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics (EACL), Association for Computational Linguistics, Morristown, NJ, USA, pp. 271–280, DOI 10.3115/979617.979652

Walker M, Fromer JC, Narayanan SS (1998a) Learning optimal dialogue strategies: A case study of a spoken dialogue agent for email. In: Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 2, Association for Computational Linguistics, pp. 1345–1351

Walker M, Litman DJ, Kamm AA, Abella A (1998b) Evaluating spoken dialogue agents with PARADISE: Two case studies. Computer Speech and Language 12(4):317–348

Walker M, Kamm C, Litman DJ (2000) Towards developing general models of usability with PARADISE. Natural Language Engineering 6(3-4):363–377, DOI 10.1017/s1351324900002503

Wiener N (1949) Extrapolation, interpolation, and smoothing of stationary time series, vol. 2. MIT press Cambridge, MA

Wilcoxon F (1945) Individual comparisons by ranking methods. Biometrics bulletin 1(6):80–83

Williams JD (2010a) At&t statistical dialog toolkit. URL http://www2.research.att.com/sw/tools/asdt/

Williams JD (2010b) Incremental partition recombination for efficient tracking of multiple dialog states. In: Proceedings of the International Conference on Acoustics Speech and Signal Processing (ICASSP), IEEE, pp. 5382–5385

Williams JD, Young SJ (2005) Scaling up POMDPs for Dialog Management: The "Summary POMDP" Method. In: Workshop on Automatic Speech Recognition and Understanding (ASRU), IEEE, pp. 177–182

Williams JD, Young SJ (2007) Partially observable Markov decision processes for spoken dialog systems. Computer Speech and Language (21):393–422

Williams JD, Poupart P, Young SJ (2005) Factored partially observable markov decision processes for dialogue management. In: 4th Workshop on Knowledge and Reasoning in Practical Dialog Systems, International Joint Conference on Artificial Intelligence (IJCAI), pp. 76–82

Young SJ, Schatzmann J, Weilhammer K, Ye H (2007) The hidden information state approach to dialog management. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, vol. 4, pp. V–149–IV–152

Young SJ, Gačić M, Keizer S, Mairesse F, Schatzmann J, Thomson B, Yu K (2010) The hidden information state model: A practical framework for POMDP-based spoken dialogue management. Computer Speech & Language 24(2):150–174

Young SJ, Gačić M, Thomson B, Williams JD (2013) POMDP-based statistical spoken dialog systems: A review. Proceedings of the IEEE 101(5):1160–1179

Zue V, Glass J, Goodine D, Leung H, Phillips M, Polifroni J, Seneff S (1991) Integration of speech recognition and natural language processing in the MIT Voyager system. In: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), IEEE, pp. 713–716

List of Contributing Publications

Parts of this dissertation have already been published in the following scientific articles:

Nothdurft F, Ultes S, Minker W (2016) User-centred spoken dialogue management. In: Next Generation Intelligent Environments, Springer, pp. 265–294

Ultes S, Kraus M, Schmitt A, Minker W (2015) Quality-adaptive spoken dialogue initiative selection and implications on reward modelling. In: Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL), ACL, pp. 374–383

Ultes S, Platero Sánchez MJ, Schmitt A, Minker W (2015) Analysis of an extended interaction quality corpus. In: Proceedings of the 6th International Workshop On Spoken Dialogue Systems (IWSDS), URL https://www.uni-ulm.de/fileadmin/website_uni_ulm/allgemein/2015_iwsds/iwsds2015_submission_4.pdf

Nothdurft F, Ultes S, Minker W (2015) Finding appropriate interaction strategies for proactive dialogue systems—an open quest. In: Proceedings of the 2nd European and the 5th Nordic Symposium on Multimodal Communication, August 6-8, 2014, Tartu, Estonia, LiU Electronic Press, 110, pp. 73–80

Schmitt A, Ultes S (2015) Interaction quality: Assessing the quality of ongoing spoken dialog interaction by experts—and how it relates to user satisfaction. Speech Communication 74:12–36, DOI 10.1016/j.specom.2015.06.003, URL http://www.sciencedirect.com/science/article/pii/S0167639315000679

Ultes S, Minker W (2014) Managing adaptive spoken dialogue for intelligent environments. Journal of Ambient Intelligence and Smart Environments 6(5):523–539, DOI 10.3233/ais-140275

Ultes S, Minker W (2014) Interaction Quality Estimation in Spoken Dialogue Systems using Hybrid-HMMs. In: Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL), Association for Computational Linguistics, pp. 208–217, URL http://www.aclweb.org/anthology/W14-4328

Sidorov M, Ultes S, Schmitt A (2014) Comparison of gender- and speaker-adaptive emotion recognition. In: International Conference on Language Resources and Evaluation (LREC), pp. 3476–3480

Sidorov M, Ultes S, Schmitt A (2014) Emotions are a personal thing: Towards speaker-adaptive emotion recognition. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pp. 4836–4840

Ultes S, Dikme H, Minker W (2014) Dialogue management for user-centered adaptive dialogue. In: Proceedings of the 5th International Workshop On Spoken Dialogue Systems (IWSDS), URL http://www.uni-ulm.de/fileadmin/website_uni_ulm/allgemein/2014_iwsds/iwsds2014_lp_ultes.pdf

Ultes S, Dikme H, Minker W (2014) First insight into quality-adaptive dialogue. In: Interational Conference on Language Resources and Evaluation (LREC), pp. 246–251.

Ultes S, Minker W (2013) Improving interaction quality recognition using error correction. In: Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue, Association for Computational Linguistics, pp. 122–126, URL http://www.aclweb.org/anthology/W/W13/W13-4018

Ultes S, Minker W (2013) HIS-OwlSpeak: A model-driven dialogue manager with multiple control modes. In: Proceedings of the 9th International Conference on Intelligent Environments (IE), IEEE, pp. 108–115

Ultes S, ElChabb R, Schmitt A, Minker W (2013) JaCHMM: A Java-based Conditioned Hidden Markov Model Library. In: International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 3213–3217

Ultes S, Minker W (2013) Interaction quality: A review. Bulletin of Siberian State Aerospace University named after academician MF Reshetnev (4):153–156, URL http://www.vestnik.sibsau.ru/images/vestnik/ves450.pdf

Ultes S, ElChabb R, Minker W (2012) Application and evaluation of a conditioned hidden Markov model for estimating interaction quality of spoken dialogue systems. In: Mariani J, Devillers L, Garnier-Rizet M, Rosset S (eds.) Proceedings of the 4th International Workshop on Spoken Language Dialog System (IWSDS), Springer, pp. 141–150

Schmitt A, Ultes S, Minker W (2012) A parameterized and annotated spoken dialog corpus of the cmu let's go bus information system. In: International Conference on Language Resources and Evaluation (LREC), pp. 3369–337

Ultes S, Schmitt A, Minker W (2012) Towards quality-adaptive spoken dialogue management. In: NAACL-HLT Workshop on Future directions and needs in the Spoken Dialog Community: Tools and Data (SDCTD 2012), Association for Computational Linguistics, Montréal, Canada, pp. 49–52, URL http://www.aclweb.org/anthology/W12-1819

Ultes S, Schmitt A, ElChabb R, Minker W (2012) Statistical modeling of Interaction Quality in spoken dialogue systems: A comparison of (conditioned) Hidden-Markov-Model-based classifiers vs. support vector machines. Bulletin of Siberian State Aerospace University named after academician MF Reshetnev, pp. 115–118

Ultes S, Heinroth T, Schmitt A, Minker W (2011) A theoretical framework for a user-centered spoken dialog manager. In: Proceedings of the Paralinguistic Information and its Integration in Spoken Dialogue Systems Workshop, Springer, pp. 241 – 246

Ultes S, Schmitt A, Minker W (2011) Attention, Sobriety Checkpoint! Can Humans Determine by Means of Voice, If Someone Is Drunk... And Can Automatic Classifiers Compete? In: Proceedings of the 12th Annual Conference of the International Speech Communication Association (INTERSPEECH), pp. 3221–32

List of Additional Publications

The following publications have not been included in the dissertation:

Pragst L, Ultes S, Kraus M, Minker W (2015) Adaptive dialogue management in the KRISTINA project for multicultural health care applications. In: Proceedings of the 19th Workshop on the Semantics and Pragmatics of Dialogue (SEMDIAL), pp. 202–203

Sidorov M, Ultes S, Schmitt A (2014) Automatic recognition of personality traits: A multimodal approach. In: Proceedings of the Workshop on Mapping Personality Traits Challenge and Workshop (MAPTRAITS), pp. 11–15

Ultes S, Schmitt A, Minker W (2013) On quality ratings for spoken dialogue systems – experts vs. users. In: Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, pp. 569–578

CV removed for online publication.