Learning User Modelling Strategies for Adaptive Referring Expression Generation in Spoken Dialogue Systems

Srinivasan Chandrasekaran Janarthanam

Doctor of Philosophy

Institute for Language, Cognition and Computation

School of Informatics

University of Edinburgh

2011

Abstract

We address the problem of dynamic user modelling for referring expression generation in spoken dialogue systems, i.e how a spoken dialogue system should choose referring expressions to refer to domain entities to users with different levels of domain expertise, whose domain knowledge is initially unknown to the system. We approach this problem using a statistical planning framework: Reinforcement Learning techniques in Markov Decision Processes (MDP).

We present a new reinforcement learning framework to learn user modelling strategies for adaptive referring expression generation (REG) in resource scarce domains (i.e. where no large corpus exists for learning). As a part of the framework, we present novel user simulation models that are sensitive to the referring expressions used by the system and are able to simulate users with different levels of domain knowledge. Such models are shown to simulate real user behaviour more closely than baseline user simulation models.

In contrast to previous approaches to user adaptive systems, we do not assume that the user's domain knowledge is available to the system before the conversation starts. We show that using a small corpus of non-adaptive dialogues it is possible to learn an adaptive user modelling policy in resource scarce domains using our framework. We also show that the learned user modelling strategies performed better in terms of adaptation than hand-coded baselines policies on both simulated and real users. With real users, the learned policy produced around 20% increase in adaptation in comparison to the best performing hand-coded adaptive baseline. We also show that adaptation to user's domain knowledge results in improving task success (99.47% for learned policy vs 84.7% for hand-coded baseline) and reducing dialogue time of the conversation (11% relative difference). This is because users found it easier to identify domain objects when the system used adaptive referring expressions during the conversations.

Acknowledgements

I am greatly indebted to my supervisor Prof. Oliver Lemon for his immense support during the past 4 years. This work wouldn't have been possible without his constant guidance. I am very grateful to him for guiding me to the right resources, helping me understand several concepts in Reinforcement Learning and Dialogue Systems and co-authoring several papers with me.

I am thankful to my external examiner, Prof. Ehud Reiter and my internal examiner, Prof. Jon Oberlander for their reviews. I thank Prof. Johanna Moore, Prof. Bonnie Webber and Prof. Steve Renals for their comments during annual reviews that helped me to carry my work in the right direction. I thank my colleagues Verena Rieser, Xingkun Liu, Paul Crook and Helen Hastie for their very useful comments and suggestions from time to time.

I thank France Telecom and Center for Speech Technology Research (CSTR) for their help with equipments required for conducting experiments with real users during data collection and evaluation.

I thank the British Council UKIERI Ph.D scholarships and the University of Edinburgh Principal's Ph.D scholarship for funding my Ph.D study. I want to thank the CLASSiC project (FP7) for funding all my experiments with real users as a part of this work.

I thank my family and my friends for their constant support and encouragement. I especially thank my wife Jay for her love, affection and amazing support during this time.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Srinivasan Chandrasekaran Janarthanam*)

To my parents..

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Spoken dialogue systems (SDS) are becoming popular and successful both in academic research and in industry with advances in speech and dialogue research. This has resulted in the development of industrial standards like Voice XML, SCXML, SRGS, SISR, PLS, SSML,[1] etc. However this success has been limited to *information seeking* dialogue tasks like getting flight or restaurant information. Industry is now venturing into using dialogue systems for collaborative problem solving in technical domains, where a system plays a role of a domain expert to help its users (Acomb et al. (2007); Boye (2007); Williams (2007)). Examples of technical domain tasks include 1) helping users with a high-functionality software application like MS-Excel, etc, 2) supporting users in troubleshooting malfunctioning gadgets like laptops, mobile phones, etc, 3) helping users install broadband Internet connections and 4) helping users learn how to use complex gadgets like digital cameras, mobile phones, etc. In such technical tasks, the system must be able to identify the domain knowledge levels of its current user and adapt its instructions accordingly. Similarly, in domains like town information where dialogue systems are deployed as tour guides to help tourists navigate the town, the system should be able to adapt to varying levels of the user's knowledge of the town when giving them directions and tour plans. As dialogue systems progress to tackle more and more complex domains, the knowledge of the domain in both dialogue partners becomes a major factor determining the success of the conversation. Therefore, in order to be natural and successful, dialogue systems should be able to use the domain communication knowledge appropriately to adapt to its users (Rambow (1990);

[1]Voice XML, State Chart XML, Speech Recognition Grammar Specification, Semantic Interpretation for Speech Recognition, Pronunciation Lexicon Specification, Speech Synthesis Markup Language, etc are standards developed by World Wide Consortium (www.w3.org) to standardize voice enabled web technologies.

Kittredge et al. (1991)).

In this thesis, we focus on one of the important dimensions of adaptation in system utterances: *referring expressions* (RE). Referring expressions are linguistic expressions that are used to refer to the domain objects of interest. Ideally, there are different ways of referring to the same domain entity and therefore such expressions have to be tailored to the user's knowledge of the task domain. A co-operative dialogue system whose goal is to provide users with instructions or information effectively must be able to choose the most appropriate expressions in its utterances. Traditionally, the referring expression generation (REG) task includes selecting the type of expression (e.g. pronouns, proper nouns, common nouns, etc), selecting attributes (e.g. colour, type, size, etc) and realising them in the form of a linguistic expression. However, in this thesis, we focus only on the user modelling aspects of referring expression generation.

User modelling and user adapted interaction has been one of the major research trends in the Artificial Intelligence community (see section 2.1). The objective of user modelling is to make interactive information systems gather information about their users and present information adapted to a variety of users. This is done by maintaining a *user model*, which stores relevant information about the user (Kobsa and Wahlster (1989); Kass (1991); McTear (1993)). User models and modelling techniques have been used in a variety of information systems like information retrieval, question-answering, dialogue systems, etc. While some systems use pre-configured static user models, others use user modelling techniques based on hand-coded rules and supervised learning methods to dynamically populate and use user models (see section 2.1.2). Dynamic user models contain information about the user that can be modified or revised by the system during the course of the conversation (McTear (1993)). In the case of dynamic models, information is obtained from users either explicitly by asking them for relevant information or implicitly inferred using rules.

In this thesis, we present a reinforcement learning (RL) approach to user modelling in a dialogue setting. We present the user modelling problem as a Markov Decision Process and use reinforcement learning algorithms to learn optimal actions to model unknown users dynamically and to adapt to them by choosing the appropriate referring expressions. We will show that a user modelling strategy can be learned from limited training data in terms of the size of the corpora and domain expertise and that such learned strategies will be able to dynamically sense the users' initial knowledge levels using unobtrusive methods and to present adapted information to them. We also show that strategies learned using our framework outperform hand-coded strategies for user modelling built using limited training data and domain expertise.

1.1 Motivation

Why must interactive systems adapt to their users? A number of studies from human-computer interaction, linguistics, science communication, psycholinguistics and educational psychology (listed below) have shown that it is indeed beneficial for interactive systems to adapt to users on various dimensions, because such adaptation increases task success and makes the interaction more natural and comfortable.

1.1.1 Human-Computer Interaction (HCI)

Usability studies suggest that systems should be able to adapt to different users with different domain expertise levels (Eberts (1994); Nielsen (1993)). Inappropriate use of referring expressions in instructions has been identified as a serious problem affecting a system's usability (Molich and Nielsen (1990)). Adaptation to users in human-computer interaction systems has been widely studied since (Carberry (1983)). User models containing information about the users provide adaptive systems with the capability to distinguish between different kinds of users and tailor their reaction based on the user's attributes (Rich (1999); Brusilovsky and Maybury (2002)). Several dialogue researchers have shown that adaptation at different levels to different attributes of the user have improved the performance of dialogue systems (Walker et al. (2004); Hassel and Hagen (2005); Winterboer and Moore (2007); Forbes-Riley and Litman (2010)). It has been shown that systems that are adaptive to users' expertise are more usable and achieve higher user satisfaction scores (Hassel and Hagen (2005)). For example, studies have shown that a system that adapts dialogue initiative strategies to the ASR error conditions at the user's end performed better than ones that did not in terms of task success rate (Litman and Pan (1999, 2002); Chu-Carroll and Nickerson (2000b)). In addition, Walker et al. (2004) show that users rate systems that tailor information to the preferences of the users much more highly than a baseline system. Similarly, Winterboer and Moore (2007) showed that user-tailored information presentation improved task success and reduced dialogue duration. Studies have also shown that tutorial dialogue systems adapting to learners' certainty in responses improves their learning gain (Forbes-Riley and Litman (2009a,b, 2010)).

1.1.2 Linguistics

Gricean maxims describe the principles of cooperative behaviour in a conversation (Grice (1975)). These maxims govern the contribution made by each interlocutor in a cooperative conversation. In particular, the *maxim of manner* suggests that contributions should be brief and orderly and that they should not be ambiguous or obscure. Therefore, the system should produce referring expressions that are not ambiguous to the users. Using expressions that the user cannot resolve because of their lack of knowledge is a kind of ambiguity. A technical expression, that is unknown to the user, is ambiguous since it can be taken to be denoting any of the domain entities in the task. Dale (1988, 1989a) suggests that one of the principles to adhere to in REG is the principle of sensitivity: REs generated should take into account the hearer's domain knowledge. In addition, Reiter (1991b,a) points out that using inappropriate expressions could confuse users and make it hard for them to understand what the system is implying. Let us consider the following examples.

1. "There is a shark in the water"
2. "There is a dangerous fish in the water"

The intention of the speaker in example 1 is not only to inform the hearer of the fact that there are sharks in the water but also to caution the hearers (or users) that it is dangerous to venture into the water. A knowledgeable user would know that sharks are dangerous and therefore this utterance perfectly communicates the speaker's intention to the user. But for those users who know nothing about sharks, one should use example 2 instead. The two utterances are not interchangeable as they would produce unwanted implications when directed to the wrong kind of user. Using the second one with knowledgeable users would confuse them as to why the system is not using the word "shark" and cause them to wonder whether there is some importance in using the words "dangerous fish" instead. The user may be led to think that the fish in the water is not a shark, but something else that he/she might not know about, which is clearly not the case here. Similarly, with naive users the former form would not produce the intended inference. Therefore, the system must produce appropriate referring expressions based on the user's domain knowledge to avoid what is called false *conversational implicatures*.

1.1.3 Technical/Science communication

Studies on technical/science communication always advise speakers/writers to analyse and adapt to the audience (McMurrey (2001)). This is called *Audience Analysis*. Paris (1988) points out that a variety of technical texts like adult encyclopedias, junior encyclopedias, manuals and text books follow different strategies to present the same subject matter. The text in adult encyclopedias is directed at an expert audience who have more knowledge than the audience for text books and junior encyclopedias. However, manuals are written for professionals who are more knowledgeable than the adult encyclopedia readers. Authors use different strategies based on the kind of modalities available to them and the kind of audience they are targeting. These observations are very much applicable to dialogue systems interacting with users with different levels of background domain knowledge. Therefore, in order to be successful, dialogue systems should also adapt to their users. For expert users, the system could use technical terms and for novice users use descriptive terms.

Similarly, during oral presentations to lay people, presenters are advised to avoid jargon and abbreviations (Lucas (2003); Beebe and Beebe (2003)). Even the content of the talk could be tailored to the audience's knowledge levels in order to make the talk interesting and useful. Although technical writing and oral presentations are not the same as technical conversations, the same principles can be applied here. Therefore, we believe that conversations, in any domain where the users' knowledge of the domain can play a part in their success, should be at the level of the user's understanding in order to be useful.

1.1.4 Psycholinguistics

Adaptations, as described above, are very natural in human conversations. In human-human conversations, dialogue partners gauge each other's domain expertise levels during the conversation using cues available from each other's utterances. Issacs and Clark (1987) show how two interlocutors adapt their language in a conversation by assessing each other's domain expertise during dialogue, by observing how they react to each other's referring expression choices. Based on their observations, dialogue partners predict the knowledge pattern of their partner and carefully adapt to their level of expertise. If their predictions are wrong, they simply use the evidence to quickly alter their predictions and adapt again for the benefit of their partner. They behave this way to maximize their chances of being understood. This process has been called *align-*

ment through Audience Design (Clark and Murphy (1982); Bell (1984); Clark (1996)). Clark and Murphy suggest that the dialogue partners adapt to each other by predicting each other's *community membership*. For instance, when a doctor talks to his new patient, he might start using simple and easy to understand words. But when he observes that his infrequent jargon is also well received by his patient and that his patient's responses contain medical jargon, he might realise that his patient is himself a medical doctor and their conversation changes to contain more jargon expressions and medical terms (Clark and Murphy (1982)). A closer look into such efficient adaptation shows that the model of the dialogue partner keeps changing dynamically during the course of the dialogue and records what his partner knows that is of interest to the current conversation. More importantly, the speaker is able to predict what else his partner might know based on what he already knows about his partner. Such adaptive behaviour is very much desirable in a dialogue system where users have different levels of background knowledge about the domain and their domain knowledge is not available to the system before the conversation starts.

1.1.5 Educational Psychology

Adapting to the user's domain knowledge is pedagogically beneficial too. Studies in *instructional science* emphasize the importance of accounting for a learner's prior knowledge in designing instructional material. Studies in Educational Psychology explain why information presented to novices can also not be presented to expert users. Expert learners tend only to benefit from complex instructions that challenge their domain skills and simplified problems or instructions tend to be beneficial to novice users only (Kalyuga (2003)). Kalyuga (2007) presents a number of empirical studies that show that complex instructions (or information) are detrimental to novices but are good for expert learners, and that simple instructions that are helpful to novices are detrimental to expert users since they increase their cognitive load considerably to process redundant information. Inappropriate levels of instruction produce wasteful cognitive load that is not useful for learning. For experts, simplified instructions and detailed worked-out steps induce unnecessary working memory load and distract them from focussing on the essentials (Renkl and Atkinson (2007); Wittwer and Renkl (2008)). Kalyuga (2009) studied two types of instruction - dynamically adaptive and non-adaptive with learners having different levels of prior knowledge. The study showed that adaptive instructions were better in terms of cognitive load, instruction time, and instructional efficiency.

Clearly, all the above studies emphasize the need for adaptation to the user's domain knowledge level so that the user is presented with appropriate information and by doing so increases task success, avoids unnecessary clarification subdialogues and repair episodes and decreases the user's cognitive load.

1.2 Challenges in dynamic user modelling

One of the important issues in adaptation is sensing the system's prior knowledge of the user. It is currently taken into account by state-of-the-art REG algorithms by querying an internal user model that has information about user's knowledge. It precisely answers the question whether the user would be able to relate the referring expression made by the system to the intended referent. The state-of-the-art REG algorithms (Dale (1988); Reiter and Dale (1992, 1995); Krahmer and Theune (2002); Krahmer et al. (2003); Belz and Varges (2007); Gatt and Belz (2008); Gatt and van Deemter (2009)) handle this problem using static user models. These user models are used to verify whether the user knows or would be able to determine whether an attribute-value pair applies to an object. So, if the user cannot associate an attribute-value pair (e.g. $< category, recliner >$) to the target entity x, then the user model would return false. On the other hand, if he can associate the pair $< category, chair >$ to x, the user model would return true. This would inform the algorithm to choose the category "chair" in order to refer to x. Therefore, using an accurate user model, an appropriate choice can be made to suit the user.

But how would a system adapt when the user's knowledge is initially unknown during run-time? In such cases, accurate user models will not be available to the system beforehand and therefore, the state-of-the-art attribute selection algorithms cannot be used in their present form. They need better user modelling to cope with unknown users. In order to deal with unknown users, a system should be able to learn about the user's domain knowledge (partially) during the course of interaction by *sensing* information about the user's knowledge and populate the user model, and to use this to *predict* the rest of the user's domain knowledge and *adapt* effectively. This is called *dynamic user modelling*. The more information the system has in its user model, the easier it is to predict the unknown information about the user and choose appropriate expressions accordingly. This is because of the fact that there are different underlying knowledge patterns for different communities of users. Novice users may know technical expressions only for commonplace domain objects. Intermediates may have knowledge of a few related concepts which form a subdomain inside a larger domain (also called as *local expertise* by Paris (1984)). Experts may know almost all the domain objects. Therefore, by knowing more about a user, the system can easily identify his/her community and more accurately predict the user's knowledge that the system is not privy to yet. There are three important steps in user modelling in this *sense-predict-adapt* approach:

1. Sensing the user's traits: How and when should the system seek information to populate the user model?

2. Adapting to the user: How can the system use an incomplete and probably a slightly inaccurate user model to predict unknown information about the user and adapt to him/her?

3. Modifying the utterance: How can the system modify its utterance to suit the user's domain knowledge based on its model of the user?

Mairesse and Walker (2010) also present a three step process to adaptation. However their focus is on the third step where they explore in detail how the adaptation parameters can be realised in the system utterances. In this thesis, we use a template based generator to handle the last step and our focus is primarily on the first two steps: sensing and adapting to the user.

One approach to sensing is to elicit information from users explicitly or implicitly in order to populate the user model dynamically during the interaction. In some dialogue systems, explicit pre-task questions about the user's knowledge level in the domain are used so that the system can produce adaptive utterances (McKeown et al. (1993)). However, it is hard to decide which subset of questions to ask in order to help prediction later even if we assume conceptual dependencies between referring expressions. Another approach is to ask users explicit questions during the conversation like "Do you know what a broadband filter is?" (Cawsey (1993)). Such measures are taken whenever inference is not possible during the conversation. It is argued that asking such explicit questions at appropriate places in the conversation makes them look less obtrusive. However, we believe that this approach is very time consuming and obtrusive for large tasks and therefore sensing should be as unobtrusive as possible.

Another issue in user modelling is to be able to use the sensed information to predict unknown facts about the user's knowledge. Rule-based and supervised learning approaches have been proposed to solve the problem of adapting to users. Rule-based approaches require domain experts to hand-code the relationship between domain concepts and rules to infer the user's knowledge of one concept when his/her knowledge of other concepts is established (Kass (1991); Cawsey (1993)). Hand-coded policies can also be designed by dialogue system designers to inform the system when to seek information in order to partially populate the user model (Cawsey (1993)). However, hand-coding adaptation policies can be difficult for large and complex tasks. Similarly, supervised learning approaches like Bayesian networks can be used to specify the relationship between different domain concepts and can be used for prediction (Akiba and Tanaka (1994); Nguyen and Do (2009)). It is also not clear how information can be sought unobtrusively from the user when using a Bayesian network approach.

While rule based approaches require domain experts to write down explicit inference rules, supervised learning approaches require large corpora of expert-lay-person interactive dialogues. In such a corpus, the expert should have exhibited adaptive behavior with users of all types. However, corpora of expert-layperson interaction or experts with both domain knowledge and experience in interaction with all kinds of users are scarce resources. Another issue is that domain experts suffer from what psychologists call *the curse of expertise* (Hinds (1999)). It means that experts have difficulties communicating with non-experts because their own expertise distorts their prediction about non-experts. Such inaccurate predictions lead to underestimating or overestimating the non-expert's capabilities. We therefore believe that, in resource scarce domains, it would be beneficial if such predictive rules for adaptation be learned using as less data as possible with little or no input from domain experts.

Our objective therefore in this study is to build a framework that can address the following two challenges:

1. Unobtrusive dynamic user modelling
2. User modelling with limited data and domain expertise

Another important point to note is that users may learn new referring expressions during the course of the interaction, and therefore the user's domain knowledge may be dynamically changing. We restrict ourselves to modelling and adapting to the initial knowledge state of the user. However, we believe that modelling and adapting to a dynamically changing user knowledge state would be an interesting extension to our current work.

1.3 Adaptive REG using Reinforcement Learning

Reinforcement Learning (RL) is a set of machine learning techniques in which the learning agent learns the optimal sequence of decisions from the feedback it gets from its environment (Kaelbling et al. (1996); Sutton and Barto (1998)). Reinforcement learning has been widely used to learn dialogue management policies that decide what dialogue action the system should take in a given dialogue state (Levin et al. (1997); Eckert et al. (1997); Williams and Young (2003); Cuayahuitl et al. (2005); Henderson et al. (2005)). A *policy* (or *strategy*) is a data structure that maps the system's various states to one of the actions that it can take in its environment. Reinforcement learning techniques are used to automatically learn policies that select the most "optimal" action in any given system state. Recently, Lemon (2008); Rieser and Lemon (2009b) have extended this approach to natural language generation (NLG) to learn NLG policies to choose the appropriate attributes and strategies in information presentation tasks. However, the application of RL for generation of referring expressions to unknown users based on user's domain knowledge has never been tried before.

Our hypothesis is that, reinforcement learning can be applied to the task of user modelling for adaptive referring expression generation where learned policies would adapt to co-operative users with different levels of domain knowledge using unobtrusive sensing methods and that such adaptation can be learned using limited resources in terms of data and domain expertise.

In this thesis, we first design the framework and build an RL agent to learn a user modelling policy from a hand-coded user simulation. We chose to study this problem in a technical support dialogue system that chooses between two kinds of expressions: technical and descriptive. *Technical expressions* (or jargon) are very specific names given to the entity and are known only to experts in the domain. *Descriptive expressions*, as the name suggests, are more descriptive and general. Although, the choices may be motivated by different reasons, we focus only on getting the user with different domain knowledge levels to identify the target entity. By domain knowledge, we mean the user's capability to identify domain objects when the system uses jargon expressions to refer to them. This is also called *domain communication knowledge* (Rambow (1990); Kittredge et al. (1991)). Therefore, this means that an expert user as defined in this thesis will not necessarily be able to reason about domain entities in terms of their functionality and how they relate with each other. It simply means that he will be able to identify the domain entities using jargon expressions.

The agent learns to adapt to users whose domain knowledge levels are unknown at the start of the conversation unobtrusively. It adapts by choosing the type of referring expression that is the most suitable to the current user. It learns the user's expertise level in the domain and adapts accordingly as the conversation proceeds. We verified the validity of our framework empirically by training the agent to learn with real user data and evaluated it with both user simulations trained using real user data, and later directly with real users. We first collected relevant data, including dialogues between real users and a "wizarded" dialogue system, using our data collection framework (see chapter 5). We then used this data to build user simulation models that simulate the real users' dialogue behaviour (see chapter 6). Using the data-driven user simulation models, we train our reinforcement learning agent to learn user modelling policies to adaptively generate referring expressions to different users dynamically. Finally, we evaluated the learned policies with simulated users and real users (see chapters 7 and 8). Figure 1.1 shows the step-by-step approach that we have followed in this study to build and validate the data-driven RL framework. Arrows represent how each step feeds into other steps in terms of design and data.

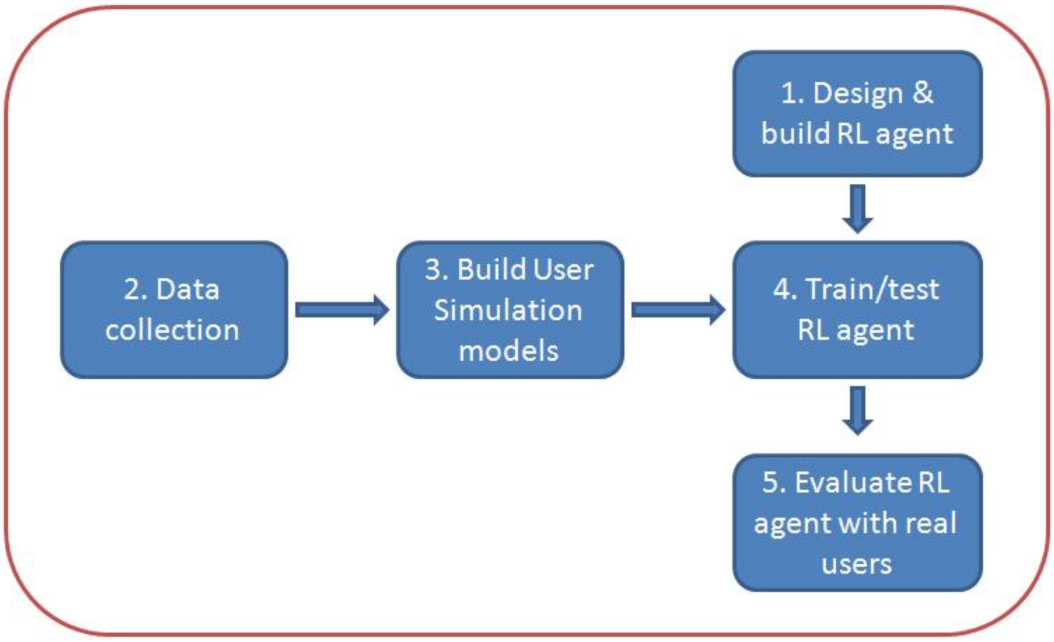

1. Design & build RL agent
2. Data collection
3. Build User Simulation models
4. Train/test RL agent
5. Evaluate RL agent with real users

Figure 1.1: Building a data-driven RL agent

1.4 Contributions

1.4.1 RL Framework for User modelling for Adaptive REG

This thesis presents a reinforcement learning framework for modelling users' knowledge unobtrusively and adapting dynamically to users with different levels of domain expertise by choosing appropriate referring expressions in spoken dialogue systems. We show how to model the user modelling problem as a Markov Decision Process. We show that, in this framework, the learning agent not only learns to senses information and adapts to users but also learns to trade off between sensing information and adapting, such that its adaptation to the user is "optimal". We also show that the agent-learned policies generalize very well to users unseen during the training phase. We validate our framework by first training and testing the agent's learned policies using data-driven user simulation models and later evaluating them on real users. We show that results from evaluation with simulated users transfer to evaluation with real users as well. We show that the learned policies adapted better than adaptive hand-coded policies in both simulated user and real user evaluation.

1.4.2 Novel user simulation models

We present new user simulation models that simulate the dialogue behaviour of users with different levels of domain knowledge levels (e.g. novices, experts, intermediates, etc.). Our models are sensitive to the referring expressions used by the system. Dialogue actions are based on the user's domain knowledge and are therefore knowledge-consistent. These models also simulate learning behaviour of real users where users learn new technical terms when engaged in a technical conversation. We first present a hand-coded simulation (in chapter 4) and later extend it to a data-driven model (in chapter 6). We show that our data-driven models simulate real users more closely than other existing models populated from the same dialogue data. We also show that the user modelling policy learned using our data-driven model outperforms our hand-coded model.

1.4.3 Bootstrapping from non-adaptive dialogue system

We also present an approach to collect a dialogue corpus with a non-adaptive dialogue system and then train the user simulation models using those non-adaptive dialogues.

This approach provides our RL framework a clear advantage over the state-of-the-art rule-based and supervised learning approaches which require domain experts to either hand-code the adaptation rules or adaptively interact with real users to create an adaptive dialogue corpus to learn from. We show that our user simulation models are designed to be responsive to the system's adaptive behavior even though they are populated from non-adaptive dialogue data.

1.4.4 Effects of adaptive REG

We show that adaptation matters significantly in a technical conversation. We also show from our experiments with real users that adaptation at the level of referring expressions does affect dialogue parameters like task success and dialogue duration. From the user satisfaction surveys conducted, we show that users found it easy to find domain entities during the conversation when using the system that adapted well to them.

1.5 Publications

The following are the publications that resulted from the work presented in this thesis.

1.5.1 Book chapters

1. Srinivasan Janarthanam and Oliver Lemon. 2010. *Learning Adaptive Referring Expressions Generation Policies for Spoken Dialogue Systems.* In Krahmer, E., Theune, M., eds.: Empirical Methods in Natural Language Generation. Volume 5980 of Lecture Notes in Computer Science. Springer, Berlin / Heidelberg (2010).

1.5.2 Conferences

1. Srinivasan Janarthanam and Oliver Lemon. 2010c. *Adaptive Referring Expression Generation in Spoken Dialogue Systems: Evaluation with Real Users.* In proceedings of the 11th Annual SIGDial Conference on Discourse and Dialogue, Tokyo.

2. Srinivasan Janarthanam and Oliver Lemon. 2010b. *Learning to Adapt to Unknown Users: Referring Expressions Generation in Spoken Dialogue Systems.* In proceedings of the 48th Annual Conference of the Association for Computational Linguistics (ACL), Uppsala.

3. Oliver Lemon, Srinivasan Janarthanam and Verena Rieser. 2010a. *Generation under uncertainty: Challenge paper.* In proceedings of the 6th International Natural Language Generation Conference (INLG), Dublin.

4. Srinivasan Janarthanam and Oliver Lemon. 2009e. *A Two-tier User Simulation Model for Reinforcement Learning of Adaptive Referring Expression Generation Policies.* In proceedings of the 10th Annual SIGDial Conference on Discourse and Dialogue, London.

1.5.3 Workshops

1. Srinivasan Janarthanam and Oliver Lemon. 2009d. *A Data-driven method for Adaptive Referring Expression Generation in Automated Dialogue Systems: Maximising Expected Utility.* In proceedings of workshop on Production of Refer-

ring Expressions (PRE): Bridging the gap between computational and empirical approaches to reference, the 31st Annual Conference of the Cognitive Science Society (CogSci), Amsterdam.

2. Srinivasan Janarthanam and Oliver Lemon. 2009c. *Learning Adaptive Referring Expression Generation Policies for Spoken Dialogue Systems using Reinforcement Learning.* In proceedings of the 13th Workshop on Semantics and Pragmatics of Dialogue (SEMDIAL), Stockholm.

3. Srinivasan Janarthanam and Oliver Lemon. 2009b. *A Wizard-of-Oz Environment to study Referring Expression Generation in a Situated Spoken Dialogue Task.* In proceedings of the 12th European Workshop on Natural Language Generation (ENLG), at 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL), Athens.

4. Srinivasan Janarthanam and Oliver Lemon. 2009a. *Learning Lexical Alignment Policies for Generating Referring Expressions for Spoken Dialogue Systems.* In proceedings of the 12th European Workshop on Natural Language Generation (ENLG), at 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL), Athens.

5. Srinivasan Janarthanam and Oliver Lemon. 2008. *User simulation for knowledge-alignment and online adaptation in Troubleshooting Dialogue Systems.* In proceedings of the 12th Workshop on Semantics and Pragmatics of Dialogue (SEMDIAL), London.

1.6 Thesis plan

In chapter 2, we review some of the previous work related to our study in this thesis. Our work relates to several subdomains of Artificial Intelligence and Natural Language Processing. We review work relating to spoken dialogue systems, referring expression generation and user modelling.

In chapter 3, we present the technical background to the Reinforcement Learning techniques and algorithms that are used in building the basic framework of this study. We also present work on how reinforcement learning techniques have been used to learn policies for dialogue management, natural language generation and language understanding.

In chapter 4, we first present an analysis of the user modelling problem and discussed why reinforcement learning is a suitable candidate to solve this user modelling problem. We have shown how to represent this problem as a Markov Decision Process. We also present our 5-step approach towards developing a data-driven adaptive dialogue system.

In chapter 5, we present the data collection framework in which we used a "wizarded" dialogue system to collect a small corpus of non-adaptive dialogues with real users. We then present an analysis of the collected data.

In chapter 6, we analyse why the state-of-the-art user simulation models are not suited for the problem at hand. We then describe in detail, the design and implementation of new data-driven models called *three-step pipeline models* that are used in this study and how these models are trained using the data that we collected. We also show that the new models simulate real user dialogue behaviour more closely than our baseline models.

In chapter 7, we describe how we retrained the learning agent using three-step user simulation models. We show that the system using the policy learned using the data-driven user simulation performed better than other baseline policies and policies learned using a hand-coded user simulation. We also show that the learned policies generalise to new users who were not seen during the training phase.

In chapter 8, we present the evaluation of the learned policies with real users in lab conditions. We show that the policy learned with the data-driven user simulation model adapted better than a baseline policy when the system interacted with real users. We also show that real users found it easier to identify domain objects when interacting with the system using the learned policy than with the baseline policy.

In chapter 9, we summarize the contributions of this thesis, and discuss applications and future research directions.

Chapter 2

Related work

We first review user modelling techniques used in various interactive systems like spoken dialogue systems, question answering systems, information retrieval systems, etc and discuss how they relate to our current problem of domain knowledge adaptation. In the next two sections, we review earlier work in user-adaptive spoken dialogue systems, natural language generation (NLG). In each of these sections, we first present a survey of different approaches and then focus on what user traits have been adapted to by these systems in the past. In the final section, we discuss previous work in referring expression generation (REG) and how our work fits into the state-of-the-art in REG.

2.1 User modelling techniques

In this section, we discuss how interactive systems represent information pertaining to users, how they are populated and how unknown information is predicted during the course of interaction. User modelling is the process of acquiring information about the users and using it to enhance the performance of the system with respect to the users. Use of user models for building adaptive interactive systems has been studied since Hayes and Rosner (1976). Kobsa and Wahlster (1989); Wahlster and Kobsa (1989); McTear (1993) survey a number of interactive systems that used user models for adaptation. Information about the user concerning his/her goals, preferences, beliefs, domain knowledge, etc is acquired, stored in user models and utilized for a variety of systems to produce user-adaptive behaviour. User models have been used in a number of information systems like information retrieval systems, recommendation systems, dialogue systems, intelligent tutoring systems, question answering systems, expert systems, adaptive hypermedia, online shopping malls, game playing, etc (Cohen and Jones (1989); Wilkinson (2006); Quarteroni and Manandhar (2006b,a); Kay et al. (2002)). In such systems, user models provide the system with essential information to adapt its output to suit the needs of the users. Similarly, user models can also be used to adapt to the user's input when it is not articulated properly or when the input is distorted in a noisy environment.

2.1.1 Representing user information

Information in user models can be represented using different kinds of models: *stereotypes*, *overlay*, *differential*, and *perturbation* models (Nguyen and Do (2008)). In stereotypical models, users can belong to one of the several predefined groups. For example, these groups can be defined based on the knowledge scale as novices, intermediates and experts (Rich (1989); Kavcic (2000)). These stereotypes define what a user knows if he belongs to that category. On the other hand, individual users' knowledge can be modelled by representing what each one of them knows instead of categorising them as novices, experts, etc. *Overlay* user models basically represent a individual user's knowledge as a subset of the domain knowledge model of the system. It can simply be a set of variables representing various domain concepts with binary values representing whether the user knows the concept or not. The *differential* model represents the learner's knowledge as an overlay on the knowledge he/she is supposed to acquire, and not the entire domain. The *perturbation* model however can store information that is not a part of the domain model. For example, it can store *mal-rules* representing the user's misconceptions (Martins et al. (2008)). In this thesis, we use an overlay model to represent the user's domain knowledge in the dialogue system. However, it is updated dynamically and is used to predict and adapt to different user types. We also use stereotypes to represent different user types in our user simulation module which is used to train the dialogue system to learn adaptive strategies for referring expression generation (see chapter 6).

Another important problem in user modelling is how user models are populated with relevant information about the user. Many systems use static user models which are populated before the interaction with the user starts. As mentioned in some of the earlier sections, many REG and NLG systems use static user models (see section 2.3.2). In other systems, user models are populated dynamically during the interaction. Systems, like GRUNDY and KNOME, that represent users' interest in terms of stereotypes, use explicit questions at the beginning of the conversation to infer the category a user belongs to and then use the appropriate user model to adapt to the user (Rich (1979, 1989); Chin (1989)). But having identified a category, the system then uses the information attributed to the stereotype to adapt to the user. However, explicit questions are considered to be intrusive as they do not directly relate to the task at hand. Cawsey (1993) also uses explicit questions like *"Do you know what a light dependent resistor is?"* to populate the user model. However, it is argued that when such questions are asked during the conversation at appropriate points, they are not intrusive. In contrast to these approaches, implicit approaches use information from user's utterances, request for clarifications, etc to populate the user model. Even systems like GRUNDY and KNOME use stereotype information once the user has been classified. On the other hand, our system acquires information about the user's domain knowledge by using jargon expressions in its instruction giving utterances during its interaction with him/her. It does not ask intrusive and explicit questions to gauge the user's knowledge of the domain. Instead the user's responses are used to infer his/her knowledge of the jargon expression. By not explicitly asking about the user's knowledge, we follow a less obtrusive approach.

2.1.2 Predictive approaches to user modelling

In addition to representing user specific information as a user model and populating the model, interactive systems need a way to predict unknown information concerning the user's interests, goals, knowledge, etc from whatever information is initially known about the user or a community of users. This can be considered as a way of implicitly acquiring more information about the user based on information the system already has about the user. For example, predicting if a user knows X, given that he knows Y. Predictive user modelling is usually done using either *content based modelling* or *collaborative modelling* or a mixture of the two. In content-based user modelling, systems adapt to users based on their interaction history. For instance, a system delivering news information on the web would be able to suggest new news items for a user based on his choices in the past (Magnini and Strapparava (2010)). This is based on the assumption that any given user repeats the same behaviour in any given (repeating) circumstance. In collaborative modelling, also called *social filtering*, users' preferences or goals are predicted based on the preferences and goals of other users. This is based on the assumption that users tend to have similar tastes as a group. Therefore by knowing a community of users who share the same rating or pattern as the current active user, a system can predict what the active user might like (Das et al. (2007)). However, these two approaches are not mutually exclusive. These two can be combined in such a way that, using content-based modelling, the active user's patterns can be used to predict his community and thereafter use collaborative modelling methods to predict his interests (Basu et al. (1998)). The user's intentions or goals and their plans to achieve them are predicted by observing a sequence of his/her actions. This process is called *plan recognition* (Schmidt et al. (1978); Carberry (1990)). However, our goal is to predict the user's knowledge of jargon expressions used in the domain. We now examine the state-of-the-art in predictive user modelling in various interactive systems.

2.1.2.1 Rule based models

One way to predict unknown information is by using inference rules (Chin (1989); Cawsey (1993)). EDGE is a system that produces an interactive explanation through dialogue to users in the domain of electronic circuits (Cawsey (1993)). The system updates the user model based on the user's clarification requests, user acknowledgements, and the system's explicit and implicit questions. The system has rules to infer what the user knows from dialogue exchanges and updates such information in the user model. The following are examples of direct inference rules.

1. If the user asks X, then the user doesn't know X.
2. If the system tells user X and the user acknowledges, then the user knows X.

The system also has a set of indirect inference rules that are used to predict a user's knowledge of which the system has no information. The following is an example of an indirect inference rule.

1. If all subconcepts are known, then the parent concept is known.

The disadvantage of this approach is that the inference rules have to be hand-coded. Also, domain expertise is required to identify how concepts are related to each other. However, rules can get more specific than the ones given above and such rules can be learned from data. *Rule induction* is a process of automatically learning sets of rules for user modelling. Basu et al. (1998) uses an inductive learning system called RIPPER that learns rules from data presented as set-valued attributes (Cohen (1995)). However, such learning requires a large amount of training data.

2.1.2.2 Linear models

Linear models predict users' preferences in order to recommend items of interest by summing up weighted attributes from the users' ratings in the past or the ratings of other similar users. Raskutti et al. (1997) uses a user's long term preferences saved in user profiles to recommend documents that may be of interest to the user (i.e. content-based modelling). Here, user profiles are created offline using a *Heuristic-Statistical* approach. A user's profile records the user's likes and dislikes by recording the features of the items they select and the qualitative ratings (e.g. like, love, hate, etc) they give them. Each feature-value pair is then rated and their ratings are accumulated and normalised, which is called *CombRating*. These are then used to build *selectional indices*, which contain conjunction/disjunction of feature-value pairs that are then used to filter items to recommend to the users. These items are then checked for compatibility with the user profile using the following linear model.

$$Compatibility = \sum Priority * CombRating$$

Priority is the priority of each attribute in the domain and is hand-coded during the implementation. Items with a high *Compatibility* score will finally be recommended to the users.

Resnick et al. (1994) predicts user scores based on the ratings of other users (i.e. collaborative user modelling). User ratings are predicted using a linear model of weighted averages of all the other user ratings on the item of interest. User A's rating of item i is predicted from the ratings of all other users, their mean ratings, and their correlation with user A (r_{AB}) using the following linear model.

$$A_i = \overline{A} + \frac{\sum_{B \in users}(B_i - \overline{B})r_{AB}}{\sum_B |r_{AB}|}$$

Linear models, such as the above, have been used in recommendation systems. These models are created using data collected from users on the features that they like and dislike. However, it is not clear how such models can be used for adaptation to domain knowledge of users. Although it might be possible to collect data on user's domain knowledge using this method, it is not sufficient to model them dynamically. In this thesis, we present an approach that learns linear models for each possible action using reinforcement learning that satisfies our objective of dynamic user modelling.

2.1.2.3 TF-IDF

The *Term Frequency - Inverse Document Frequency (TF-IDF)* model is a user modelling technique that has been used extensively in information retrieval in which each document in the corpus is represented by a vector of weights, with each weight representing a word in the document. Cosine similarity between the user's query vector and the document vector is then used for recommendation (Salton and McGill (1983)). Balabanovic (1998); Moukas and Maes (1998) used TF-IDF based methods to recommend documents to users based on their interests. Balabanovic (1998) presents an interesting way to model user's interests for information retrieval. In this method, a set of documents D is represented using a n-dimensional vector ($\{d_1, d_2 ... d_n\}$). Each document d_i is represented by a set of values for p keywords ($\{t_1, t_2 ... t_p\}$). Each element of this vector is derived by multiplying a term frequency (TF) component, which is the number of times the word j appears in the document d_i, by an inverse document frequency (IDF) component, which is the inverse of the number of documents in the corpus in which the word j appears.

The system represents each user with a user model. The user model (q) is also represented as a p-dimensional vector. This contains an array of weights $\{w_1, w_2 ... w_p\}$ corresponding to the keywords. The transpose of q is denoted by q^T. User adaptation is then done using a linear preference function $f(d)$ as follows.

$$f(d_i) = q^T t = \sum_{1 \leq j \leq p} w_j t_j$$

The above function returns a preference value for each of the documents in the set D which helps the system in presenting the user with the most preferred document. User models such as these can be learned using a simple gradient descent procedure (Wong and Yao (1990)). The disadvantage of this method is that it cannot be directly applied to our user modelling problem for adaptive REG as adaptation to a user's knowledge is not the same as the adaptive information retrieval problem.

2.1.2.4 Markov models

Markov models have been widely used for user modelling. Bestavros (1996); Horvitz et al. (1998); Zukerman et al. (1999) use Markov models based on the Markov assumption, according to which the occurrence of the next event is based only on previously occurring event. This is a collaborative approach to user modelling, in that, the current user's choice of document is predicted based on the other users' choices in the past.

Zukerman et al. (1999) use Markov models to predict the most likely document ($P(D)$) that the user might ask for given the user's request history (H). This helps the remote server to send webpages or documents that are highly likely to be requested before they are even requested, with the objective of reducing the user wait time. Such a model is called a *Time Markov* model and can be modelled as a conditional probability as shown below.

$$P(D|H)$$

Similarly, a *Space Markov* model predicts the next document based on the referring document that has a link to the next document. A *Linked Space-Time Markov* model predicts based on both the previously requested document and the referring document. These probability models are built using a large corpus of user requests. Although, Markov models can also be used to predict a user's knowledge of technical terms using incomplete information about their knowledge, it still doesn't solve our problem of dynamic modelling. For instance, how can user models be populated at the beginning of the conversation?

2.1.2.5 Clustering models

Some studies use clustering algorithms to cluster similar web pages together so that related webpages can be presented to users based on their visiting patterns. Perkowitz and Etzioni (2000) presents *Page Gather* algorithm which takes as input the website logs. Using a variant of a traditional clustering algorithm called *Cluster Mining*, it clusters similar webpages together using co-occurence frequencies and creates an index page for each cluster. Based on the current user's visit pattern, appropriate index pages can be presented to them. One of the main advantages in clustering approaches is that they do not require a labelled corpus of training data like supervised methods. However, this type of adaptation can be used only for document retrieval problems.

2.1.2.6 Bayesian networks

Bayesian Networks are directed acyclic graphs with nodes representing random variables and arrows between them representing causal or influential links from parent nodes to child nodes (Pearl (1988)). Each node has a conditional probability distribution which defines the probability of the random variable taking different values given the values of its parent nodes. Bayesian networks support both collaborative and content-based modelling. The conditional probability distributions can be learned using collaborative methods from a corpus of user interactions and can be used with individual users using content-based modelling to set the prior probabilities. These probabilities can then be used for inference and prediction of unknown information about the user. Several studies have used Bayesian networks for predictive user modelling (Akiba and Tanaka (1994); Jameson (1995); Horvitz et al. (1998); Albrecht et al. (1998); Horvitz et al. (1999); Jameson et al. (2000)).

Recently, Nguyen and Do (2009) proposed a combination of Bayesian networks with the overlay model to predict the learner's knowledge in a tutorial domain. In this study, the learner's knowledge of domain concepts is represented as hidden variables and the learner's activities that serve as evidence to his knowledge (e.g. his performance in assignments/tests) are represented as evidence variables of the network. These variables are binary and represent the two states of whether the learner knows the domain concepts. The domain concepts are places in a hierarchy of prerequisites (parent-child relationship in the network) representing what the learner should already know in order to master a larger concept. For example, in order to know "Java", the learner must know "Control structure", "Class & Object" and "Interface" concepts. Each of these prerequisites are weighted such that the sum of all the weights on different parents on a child add up to 1. Finally, the conditional probability distribution of every node is defined by building conditional probability tables using the following formula.

$$P(X = 1|Y_1, Y_2 ... Y_n) = \sum_{i=1}^{n} w_i * h_i$$
$$\text{where } h_i = \{1 \; if \; Y_i \; = \; 1, \; 0 \; otherwise\}$$

The conditional probability tables are then used predict the learner's knowledge of concepts (represented by hidden nodes) based on the available information (represented by evidence nodes). This process is called *knowledge diagnosis*. Although Bayesian networks provide an elegant way to represent a user model and reason with it under uncertainty, the relationships between the nodes have to defined by domain experts. Nguyen and Do (2009) also notes that the disadvantage of Bayesian networks is that it requires a large data storage for the probability tables as the network becomes larger and more complex and that the computation of the posterior probability of the hidden nodes takes considerable amount of time. Another disadvantage is that, similar to Markov models, Bayesian Network models cannot be used for dynamic modelling without another mechanism to populate the input nodes at the beginning of the conversation.

In summary, we have reviewed several user modelling techniques used in user-adaptive systems. We argue that some of these, like TF-IDF and clustering models, are applicable only to information retrieval tasks. However, others can be tailored to solve our problem of adapting to user's domain knowledge. We also argue that techniques like Markov models and Bayesian Network models cannot be applied directly without a mechanism to first partially populate the user model using which inference can be done. In contrast to these techniques, we implement a combination of reinforcement learning and linear models to both adapt to user's domain knowledge and do so dynamically during the conversation.

2.2 Spoken Dialogue Systems

Spoken Dialogue systems (SDS) are human-computer interfaces that converse with a human user in order to complete a task or solve a problem. These systems provide the user with a very natural means of interaction with a computer. Over the past years, several task specific dialogue systems have been built. There are systems that provide flight schedule information (Seneff and Polifroni (2000); Levin et al. (2000); Rudnicky et al. (2000)), town information (Johnston et al. (2002); Lemon et al. (2006)), bus information (Raux et al. (2003, 2005)), weather reports (Zue et al. (2000)), etc. Dialogue systems are also used as in-car applications for music track selection (Hassel and Hagen (2005); Becker et al. (2006)), news and route advice (Rogers et al. (2000)), etc. Dialogue interfaces are also finding their way in to multi-modal systems that include other modalities like gestures and GUIs (Lemon et al. (2001); Lopez-Cozar et al. (2005)). Dialogue interfaces have also been widely used in intelligent tutoring systems that interact with students to help them solve problems and learn during the process (Graesser et al. (1999); Litman and Silliman (2004); Jordan et al. (2006); Callaway et al. (2007)). Recently, the focus has shifted to more complex domains like *Self-Help*, where the system engages the user in technical tasks like troubleshooting (Acomb et al. (2007); Boye (2007); Williams (2007)). For instance, the system could help the user troubleshoot his broadband connection by requesting information and providing instructions.

A standard architecture of dialogue systems is shown in figure 2.1. The user's utterance (as acoustic signals) are translated to a stream of words by the automatic speech

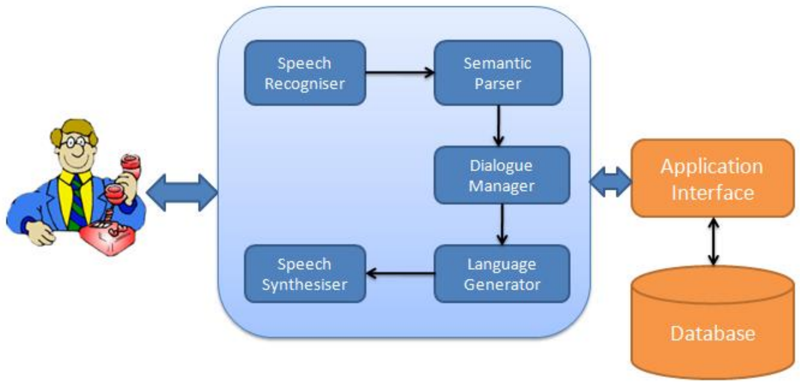


Figure 2.1: Spoken Dialogue System - Architecture

recogniser module. The semantic parser is a natural language understanding (NLU) module converts the stream of words into semantic frames called dialogue actions which are meaning representations of the user's utterances. The user dialogue actions are analysed by the dialogue manager (DM) and an appropriate response is formulated in consultation with the current state of the dialogue and the backend application. The system dialogue action produced by the dialogue manager is translated into utterance form by the natural language generator (NLG) module and then is converted into acoustic signals by the speech synthesizer (or text-to-speech (TTS)) module.

But, designing a dialogue system is more than putting together these modules. Although these modules are necessary, what makes a dialogue system natural and effective is its ability to coordinate these modules in a natural conversation with its dialogue partner. This task is the responsibility of the dialogue manager (DM). It manages the conversation using a plan, called the *dialogue policy* (or *dialogue strategy*) that maps any dialogue state to a dialogue action. The dialogue state maintains the system's knowledge, beliefs and observations of its environment (i.e. its current user), dialogue history, goals, etc. An enriched state may also contain the modality of interaction, and the user's profile containing his level of expertise, cooperativeness, etc. Actions that the DM can select (collectively called the action set) include for example, greeting the user, requesting more information, presenting the results of the task, confirming existing information, closing the dialogue, etc. Dialogue policies can be manually coded for different tasks and situations. But when there is a large number of factors affecting the dialogue (i.e. larger state space), manual coding can become difficult. One of the solutions to this problem is to learn the policies from human-human or human-machine dialogue data. Reinforcement learning has been widely used to learn dialogue management policies (see section 3.5.1 for a brief discussion).

2.2.1 Adaptive Spoken Dialogue Systems

User-adaptation has been identified as an important attribute that contributes to the success of dialogue systems. In the past, several spoken dialogue systems have been developed with user-adaptive features. In the following sections, we discuss the user traits that these systems have been designed to adapt to.

2.2.1.1 User's speech patterns

User models based on a user's speech patterns can be used to improve the performance of automatic speech recognition modules of dialogue systems. Information on the user's barge-in patterns and how accurately his/her speech has been decoded in the past (ASR accuracy) have been used to improve the performance of the system. Komatani and Okuno (2010) present a method to automatically classify correct and incorrect barge-ins using user models. When barge-in utterances from users are detected, the system stops its prompt and starts listening to the user. Hence, when background noise is incorrectly interpreted as a barge-in, the system incorrectly stops its prompt. By using user models containing information on user's ASR accuracy and barge-in rates from previous conversations together with the current ASR confidence scores, it is possible to accurately identify false barge-ins and respond appropriately. Evaluation on an annotated corpus of 7000 barge-in utterances showed that this approach of using a user model containing information on ASR accuracy measures and barge-in rates along with ASR confidence scores classifies barge-ins more accurately (92.6%) than other approaches.

2.2.1.2 User's system skills

Chu-Carroll (2000); Chu-Carroll and Nickerson (2000a) present MIMIC, a dialogue system that adaptively chooses initiative strategies during the conversation based on the user's skills in handling the dialogue system. The system takes initiative in a conversation with a novice user whereas it behaves as a passive partner with an expert user. The user's dialogue behaviour is analysed based on discourse and analytical cues of their utterances and based on this information, the system responds to the user appropriately. For instance, the system gives the user a sub-task in order to take the conversation forward when the user himself is not forthcoming with any new information.

Evaluation with real users showed that the adaptive version of the system performed better than its non-adaptive counterparts in terms of task success and user satisfaction.

Hassel and Hagen (2005) present an in-car dialogue system that adapts its language to users' skill levels in handling the dialogue system. While for experts, the prompts are generally short and terse, for novices they are exhaustive, giving all possible options the users can say, etc. Adaptation is done unobtrusively in the sense that the system evaluates the user for his skill level after every dialogue session, which is used for adaptation in future interactions. Similarly, AthosMail, a speech based interactive e-mail application, adaptively interacts with users based on their system skills (Jokinen and Kanto (2004); Jokinen (2006)). Adaptation is done based on two user models: online and offline. The offline model evaluates the user's expertise at the end of each dialogue and feeds it to the online model. The online model is updated during the dialogue and allows the system to adapt to variations found in the immediate context during runtime. Parameters like time-outs, help requests, interruptions and speech recognition problems are used to judge the user's expertise. Based on this judgement, appropriate responses are chosen. Three levels of responses are available to choose from based on the skill level of the user. Responses are detailed for novices. Unnecessary excessive information is removed and only core information is presented as expertise increases.

Komatani et al. (2003, 2005) present a learning approach to user adaptation. Using a corpus of dialogue data, a decision tree is learned to classify users based on their skills in using the dialogue system. The classification is done on dialogue features like user's silence, barge-ins, recognition scores, etc. The decision tree is used after every utterance to update the user model and issue responses based on the user model. Although the system also adapts to the skills of the user, the information about the user's skills is updated only at the end of the dialogue to be used in subsequent dialogues with the same user. As with other systems, the responses are short for hasty, expert users and are detailed for others. In contrast to this approach, our system updates the user model during the conversation and adapts dynamically.

2.2.1.3 Users' goals and preferences

In many spoken dialogue systems that present information from databases to users, there is a common problem of how the results can be presented so that they are useful to the user, and how they must be organized so that they are easy to understand and memorable to the user. GRUNDY is a dialogue system that plays the role of a librarian

(Rich (1979)). It interacts with users using a typed natural language messages and recommends books that users might like based on their interests and preferences. It uses a user modelling component that stereotypes users into many classes (e.g. sports-person, feminist, etc). It uses explicit questions to classify the user and then uses stereotype information to present information to the user. Carberry et al. (1999) also present a rule-based approach to modelling user preferences. Rogers et al. (2000) present an in-car dialogue system that advises users on routes to take, gives town information, and reads news. The system adapts to the user's task based preferences which it learns automatically from the user's choices in previous interactions. The system keeps updating the user model using simple heuristics to modify the user model. For example, if the user listens to the whole story, it is considered a positive feedback and when a new story is interrupted it is considered a negative feedback. It uses the updated models to choose appropriate content that is interesting or useful to the user.

Carenini and Moore (2001) present a framework for producing *evaluative arguments* using user models based on *multi-attribute decision theoretic* models. They showed that arguments that were tailored to user's preferences produced more effective arguments than the non-tailored version. Moore et al. (2004) present an adaptive information presentation module called FLIGHTS (Fancy Linguistically Informed Generation of Highly Tailored Speech) which presents flight information to users based on their preferences. Three sample user profiles: student, frequent flyer and business class flyer were used to tailor the system utterances at a number of levels like content selection, referring expression generation, aggregation, discourse cues and the use of appropriate scalar terms. *Multi-attribute decision* models were used to represent the user's preferences (Carenini and Moore (2001); Walker et al. (2002, 2004)). These models are developed by asking the users to rank a list of attributes (e.g. fare-class, layover-airport, etc) and specify preferred and dispreferred attributes. Using these models for different users and measures of compellingness, the results from the database search can be filtered so that only those flights that suit the users' preferences are selected. Attributes to describe were also then selected on how compelling they are to be presented to the user. Finally, a planning agent is used to produce an information presentation strategy which decides how to group and order the options and the attributes, choose referring expressions, decide how to contrast between options and decompose the strategies into dialogue acts and rhetorical relations.

In a similar study in the *restaurant* domain, Walker et al. (2004) showed that users rate the responses generated using their own models more highly than ones generated using a random user model. Later, Demberg and Moore (2006) extended information presentation in dialogue systems to handle large numbers of options by building a cluster-based tree structure to rank the options based on user preferences for stepwise refinement. Winterboer and Moore (2007) found this approach to be more effective than the *summarize and refine* approach presented in Polifroni et al. (2003). It was found that in a dual task environment (with one task demanding huge cognitive load), the user-model tailored presentations with stepwise refinement were more effective in terms of both task success and dialogue duration, although it did not get better user satisfaction ratings than the baseline system. Recently, Rieser and Lemon (2009b) presented a reinforcement learning based approach to information presentation in which the system learns to choose the number of attributes to present and the strategy to be used (e.g. summarize, compare, etc). We discuss this work further in section 3.5.3.

2.2.1.4 Users' domain knowledge

Users' domain knowledge is also one of the user traits that is modelled by interactive systems in order to present useful and adapted information to users. UNIX Consultant (UC) is an interactive dialogue system that answers questions about the UNIX operating system (Wilensky et al. (1984); Chin (1986)). It advises users on queries about various commands used in an UNIX environment (e.g. rm, vim, etc). The UC system uses a user modelling module called KNOME (Chin (1989)). It represents users' knowledge of UNIX as stereotypes using which UC tailors its response to its users. It tries to not present information that the user already knows. It is also used to understand a user's problem from a proper perspective. For instance, if an expert or an intermediate user asks how to delete a file (which even a beginner would know), the system should try and understand if there are additional problems the user is trying to solve. UC represents users using stereotypes at different levels of domain knowledge: novices, beginners, intermediates and experts. It also classifies the knowledge components in terms of difficulty: simple, mundane, complex and esoteric. It is assumed that novices know at most simple facts, intermediates know all simple, most mundane and few complex facts, and so on. The system starts off with an assumption that the user it is interacting with is a beginner. It then deduces the category to which a user belongs to by examining the statements he makes during the course of his interaction with the system. Based on whether the user knows a simple, mundane, complex or esoteric concept, the system deduces the user's category based on hand-coded rules. It eliminates categories based on the rules to select the stereotype that best matches the current user. However, once a stereotype is selected, it is not possible to change it in the face of conflicting evidence. Hand-coding rules to classify users to different sterotypes and classify concepts to different levels of difficulty requires significant manual effort from domain experts. UC uses stereotypes and classifies users dynamically through a process of elimination into a category using hand-coded rules. In contrast, our system does not classify users but predicts user's domain knowledge based on a user modelling policy that it has learned from other (simulated) users.

COMET (Coordinated Multimedia Explanation Testbed) is a multimodal dialogue system that interacts with users in a technical domain task (McKeown et al. (1993)). An important feature of this system that is relevant to this thesis is that it is able to modify its utterances to different users. The objective of this system is to only use words that the user knows when there are no accompanying pictures to disambiguate or facilitate interpretation. Using different words in place of each other may not be as easy as it seems. For instance, its not easy to replace the word "polarity" with a simpler word in the utterance "Check the polarity". Instead, it rather needs an entire overhaul of the utterance by means of rephrasing it in simpler language. E.g. "Make sure that the plus on the battery lines up with the plus on the battery compartment". The choice of words not only affects the other words in the utterance but also the subsequent utterances. Therefore the words need to be chosen with utmost care.

The lexical choice module of the text generator component of the system that generates the utterances selects words using the following four strategies.

1. Alternative words: Selecting alternative words that fit in instead of a complex one (e.g. "some number" instead of "arbitrary number").

2. Conceptual definitions: Use conceptual definitions to rephrase the utterance (like the "polarity" example).

3. Descriptive expressions: Use descriptive expressions instead of technical ones (e.g. "cable that connects to KY57" instead of "COMSEC cable").

4. Anaphoric expressions: Use anaphora to refer to entities mentioned in the past discourse (e.g. "the cable that you just removed" instead of "COMSEC cable").

The lexical choice module chooses the words based on several constraints: syntactic, semantic, discourse, and user model. Initially, the lexicon is consulted to retrieve the appropriate rules to translate the semantic constraints in the given logical form to lexical and syntactic features. These are then translated into utterances by choosing appropriate words that satisfy the pragmatic and user model constraints. If none of the readings produced are intelligible to the user, the utterance is rephrased. The lexical choice module interleaves with the content planner to choose appropriate words/phrases and therefore replan the content of the whole utterance when user model constraints are not satisfied. The user model indicates the domain knowledge level of the user in terms of his word preferences, known abbreviations and technical terms. Although the system adapts to users at different levels of domain expertise, it is to be noted that an accurate model of the user's expertise is made known to the system before the conversation starts. Like COMET, our objective is to build a technical support dialogue system that adapts to users with different levels of domain knowledge. However, unlike COMET, we assume that the user's knowledge level is unknown to the dialogue system at the beginning of the conversation. Although there are several levels of adaptation to users based on their domain knowledge, we focus on the use of appropriate referring expressions.

2.3 Natural Language Generation systems

Natural Language Generation (NLG) is the process of converting semantic and other forms of information into their equivalent textual form. NLG systems are used in a number of applications including generation of weather forecasts (Goldberg et al. (1994)), summarizing statistical data (Iordanskaja et al. (1992)), producing responses to the user in dialogue contexts, generating answers in question answering (QA) systems (Reiter et al. (1995); Paris (1987)), explaining the reasoning of an expert system (William (1983)), etc. These modules are used either as independent systems or as a part of larger interactive systems like dialogue systems, QA systems, etc. In the last section, we discussed adaptive NLG modules used in dialogue systems. In this section, we focus on independent NLG systems and NLG modules in other interactive systems.

2.3.1 Approaches to NLG

First, let us briefly explain the language generation processes in general before . There are several ways to generate text in information systems.

Template: The simplest method is to use templates, strings or syntactic trees with slots to be filled during processing. The appropriate template is retrieved and its slots are filled to produce a final text (van Deemter and Odijk (1997); White and Caldwell

(1998); Busemann and Horacek (1998); Theune et al. (2001); McRoy et al. (2003)). For example, a template with unfilled slots (e.g. \$n, \$departure_city) and an utterance with slots filled in with information is given below.

Template: "There are \$n flights from \$departure_city to \$arrival_city."
Example: "There are 5 flights from Edinburgh to London Heathrow."

Pipeline approaches: When there is a need for more flexibility in language generation, a pipeline architecture is used. It generates language in several steps: *content determination*, *discourse planning*, *sentence aggregation*, *lexicalisation*, *referring expression generation* and *linguistic realisation* (Reiter and Dale (1997, 2000)). In the first step, the system decides what information to communicate to the user. In the discourse planning step, the order and structure of the text are decided. In the sentence aggregation step, the system structures the information as a sequence of sentences. It also tries to aggregate information into fewer sentences if possible. Finally, in the lexicalisation step, the system decides what words or phrases to use in order to express the information that is to be conveyed to the user. For a detailed explanation see Reiter and Dale (1997, 2000). NLG systems like Joyce (Rambow (1990)), PERSONAGE (Mairesse and Walker (2007, 2008, 2010)) have been built based on the pipeline architecture.

Planning approaches: One of the earliest works in using planning for natural language generation was done by Cohen and Perrault (1979). Koller and Stone (2007); Garoufi and Koller (2010) present a planning approach to generation of LTAG (Lexicalised Tree Adjoining Grammars) trees. The sequence of actions consisting of substitutions and adjunctions from the top most node to build a grammatically complete tree from a semantic representation is seen as a sequence of planning actions. The syntactic and semantic constraints are encoded as preconditions and subtree they build are encoded as effects of these actions. Classical planning algorithms are then used to build a plan which is decoded into a TAG derivation.

Statistical approaches: Langkilde and Knight (1998) present Nitrogen, a two-step architecture that combined both symbolic and statistical methods towards natural language generation. The architecture uses two steps: *generation* and *extraction*. The generation step produces a word lattice of all possible renderings for the given semantic input. This is done using symbolic databases containing morphological and grammatical knowledge. In the extraction step, the system extracts the most fluent path through the word lattice. This is done by statistically ranking all the possible paths through the word lattice using bigram and unigram statistics from a large corpus. Oberlander and Brew (2000) proposed a variant of Nitrogen in which the generation (i.e. the first step) was also a stochastic process. Oh and Rudnicky (2002) presented a statistical approach for both content planning and a surface realisation. Models for both subtasks were trained on a corpus of travel reservation dialogues. The utterance is generated in a three step process of first selecting the number of attributes to include given the utterance class (e.g. inform_flight, query_depart_date, etc), second identifying the attributes to present, and third choosing the word sequence to use (surface realisation). Each of these steps were modelled using probabilistic models trained using the corpus. The candidate utterances produced were scored using a penalty score based on heuristics like length of the utterance, repetition of information, presence of invalid information, absence of valid information, etc and the one with the lowest score or zero was presented to the user. Duboue and McKeown (2003); Barzilay and Lapata (2005) present approaches to learning content selection rules from a corpus. Walker et al. (2007) presents SPARKY, a sentence plan generator and ranker that produces plans and ranks the alternative realizations. The ranker is trained on corpus data containing human rankings. A clear advantage of this approach is being able to train a NL generator from a human-human corpus and therefore produce utterances mimicking that of the humans. However, supervised learning approaches to train statistical models requires large amounts of data to train and can only learn strategies that are available in the data.

2.3.2 User-adaptive NLG

While we are not developing an alternative approach to NLG, we focus on user modelling which is an important problem in the development of user-adaptive NLG systems. Several systems have tried to present information to users in an adaptive fashion taking into account the user's interests, goals, knowledge and so on. We already discussed some adaptive NLG modules in the context of interactive dialogue systems in Section 2.2.1. In the following section, we present some NLG systems that are adaptive to a user's personality and domain knowledge.

2.3.2.1 Adapting to user's personality

Mairesse and Walker (2007, 2008, 2010) present PERSONAGE, a parameterizable personality based natural language generation (NLG) module. This system can generate language displaying different personality styles based on the "Big Five" model of personality traits (Digman (1990)). PERSONAGE can generate system utterances for recommendation and comparison of restaurants. Such utterances can be tailored to different personalities based on different scales like *extroversion* (i.e. dominance versus submissiveness), *emotional stability* (nervous vs confident), *agreeableness* (compassionate and cooperative vs suspicious and antagonistic), *conscientiousness* (organized vs carelessness) and *openness to experience* (curious vs cautious). Evaluation of utterances produced by PERSONAGE by human judges showed that the utterances were moderately natural with a mean score of 4.59 out of 7. In future, PERSONAGE could be used in interactive dialogue systems to adapt utterances to a user's personality.

2.3.2.2 Adapting to user's knowledge

In this section, we discuss four NLG systems: ROMPER, EPICURE, TAILOR, and M-PIRO, which adapt to users' domain knowledge. ROMPER is a system that provides clarification to users' misconceptions about the domain using a user model to tailor its responses to their domain knowledge (McCoy (1985, 1989)). It deals with two kinds of misconceptions: misclassification and misattribution. An example of misclassification would be for a user to think that whales are fish. In such a case, ROMPER offers the user an explanation containing different pieces of information such as whales are not fish (denial), whales and fish are similar in some aspects (possible reason for misclassification), whales are mammals (correct classification), whales breathe using lungs and not gills (reason behind correct classification) and so on. Appropriate bits of information are chosen based on a user model that informs the system of the user's knowledge structure that led to the misconception in the first place. The output of ROMPER is a formal specification of the explanation which is then converted to actual text using a realiser called MUMBLE (McDonald (1980). ROMPER uses hand-coded rules to select appropriate schemas for explanation based on the misconception and the user model. Also, it is not clear how the user models are populated in the first place, which is an issue that we focus upon in this thesis.

EPICURE is an NLG system that adapts the complexity of instructions in presenting cookery recipes according to the user's domain knowledge (Dale (1989a,b)).

EPICURE is similar to our system in the sense that our system produces a technical recipe for setting up a broadband Internet connection at home. EPICURE starts with discourse planning where plans are retrieved from a plan library to satisfy the top-level goal (e.g. making butterbean soup). The plan is recursively decomposed into simpler plans or primitive operations (or domain actions) that are known to the user. The detailed plan structure is used to produce the discourse structure. Besides discourse planning, the system pays much attention to generating referring expressions for domain objects which by themselves undergo changes as one reads the recipe. For instance, carrots or onions that are countable nouns before grating or chopping actions then become mass nouns. We will focus on user adaptive aspects of the system that are of interest to this thesis. The system stores information concerning the user it is interacting with in a user model. It consists of three kinds of information:

1. Knowledge of domain actions: Whether the user knows how to carry out domain specific actions (e.g. prepare the beans).

2. Knowledge of domain entities: Whether the user knows and would identify domain entities (e.g. kumquats).

3. Knowledge of the taxonomy of domain entities/actions: How entities/actions relate to each other (e.g. "salt" is an ingredient of "seasoning")

The user model is applied extensively in discourse planning. Complex actions (or high-level actions) that the user does not know how to execute are in turn decomposed into simplified sequences recursively until the whole plan is compatible with the given user model. The user adapted discourse plan is then sent for realisation.

TAILOR is a natural language generation component of a question answering system that provides access to a large knowledge base of information (Paris (1987, 1988)). Such systems become more usable if they tailor their responses to the user's domain knowledge levels. The main objective of the TAILOR module is to produce descriptions of domain objects tailored to the level of the domain expertise of the user. The module employs user models that represent the domain knowledge levels of the users. The models contain two kinds of information: whether the user knows the underlying basic concepts of the domain and the list of domain objects that they know in terms of their use and functionality. The module uses two strategies: *constituency schema* and *process trace*. A constituency schema (McKeown (1985)) is a textual structure that describes a domain object in terms of its subparts. And a process trace is a textual structure that describes the functionality of the domain object in terms of the mechanical process and causal links involved in the functioning of the domain object. The system produces more process trace type information to novice users who do not know even basic information as to how domain objects interact with each other. With expert users, the system produces more constituency schema type information assuming that they will be able to mentally construct the process information when new information about subparts are presented to them. The two strategies are combined for intermediate users who have local knowledge of some domain objects but not others.

The following illustration explains how the system switches between the two strategies and finds a balance between the two according to the user model. Let us suppose that the system is requested to generate a description of a telephone. Futher assume that the user model hints at the fact that the user is an intermediate user since he knows something about the domain. Therefore, it uses a constituency schema to describe the subparts of a telephone: transmitter and receiver. It then describes how the transmitter works, which is a process trace. However it only uses the constituent structure for the receiver, as the user already knows about loudspeakers. In this way, the system mixes the two different strategies to provide the user with useful information, and at the same time avoid redundant information. However, as with the previous systems, TAILOR assumes the availability of an accurate user model to support its text planning activity.

ILEX is a NLG system that produces natural language descriptions of artifacts in a museum (O'Donnell et al. (2001)). It produces descriptions that are tailored to the user's interests. The description of the artifact that the user is currently viewing would be personalised based on the artifacts that the user has already seen. However, this system does not distinguish between user types in terms of their domain knowledge. This system was followed up with M-PIRO, which allowed the domain authors to define one or more user types (Androutsopoulos et al. (2001); Isard et al. (2003)). Each entity (abstract or physical) in the database is annotated with specific information with regard to various user types. The database therefore contains information about how important or interesting each artifact is to different user types. This information, which is populated with the help of museum guides, domain experts and curators, is then used to generate descriptions adapted to the current user.

The ROMPER system deals with misconceptions in a user's domain knowledge. However, we do not study misconceptions at this stage of research. We might however extend our work to learning strategies to deal with users' misconceptions as well. EPICURE and TAILOR use user models that are static containing information on users' domain knowledge. M-PIRO on the other hand uses stereotypes which can be set or reset when the interaction starts. In contrast to these approaches, in our system we use a dynamic user model that records the user's expertise in the domain during the course of the conversation. In contrast to these systems where the association between the user models and the kind of information presented to the user is done manually by domain experts, we learn such associations from interactions with simulated users. Although, we focus only on adaptation at the level of referring expressions, we hypothesize that our framework can be used to learn user modelling strategies for all stages of NLG such as content determination, sentence planning, etc.

2.4 Referring Expression Generation

In this section, we present how our work relates to the current state-of-the-art in referring expression generation. First, we present general approaches to traditional referring expression generation (REG) problems and later discuss adaptive generation of referring expressions based on users' domain knowledge. Referring Expression Generation is a sub-process of natural language generation (NLG) which identifies the most appropriate linguistic form to refer to domain entities or eventualities (Reiter and Dale (2000); Oberlander and Dale (1991)). The choices range from noun phrases (e.g "the blue chair"), pronouns (e.g. "it"), one-anaphora (e.g. "the blue one"), etc. A classic problem that has been addressed by a number of researchers is to determine which attributes should contribute to the content of definite noun phrases that refer to a domain entity. This is called the *content determination* or *attribute selection* problem. The entity to be referred to is called the *target entity*. It is usually assumed to be present in a context of several other entities called *distractors*. The task of the REG algorithm is to then find attributes that will identify the target entity uniquely from that of the distractors. Attributes or properties of the entities can either be absolute like colour, size, orientation, and so on, or relative to other entities in the scene (e.g. "next to", "on top of").

Some work in REG focusses on how to generate expressions that satisfy multiple goals (Appelt (1985)). Referring expressions can be chosen in such a way that they carry out more actions than just identifying the referent to the hearer. For instance, an utterance like, "Could you get me the wheelpuller?" along with a pointing action towards the object not only requests the hearer to help the speaker reaching the object but also informs the ignorant hearer that the strange object being pointed to is called "a wheelpuller". One may use words like "genius" instead of "man" to not only point to the person but also display the speaker's emotion (e.g. reverence or admiration) towards the referent. Poesio et al. (1999) presented an approach to generate *non-referring parts* of referring expressions. Non-referring parts are added to expressions to communicate additional information about the referent to the user. However, unlike these two approaches that produce referring expressions to satisfy multiple goals, our work focuses on generating referring expressions for the sole objective of enabling the hearer to identify the referred entity.

2.4.1 Approaches to REG

2.4.1.1 Rule-based approaches

The *incremental algorithm* is a popular attribute selection algorithm that selects attributes in a predetermined order incrementally (Reiter and Dale (1992, 1995)). This is motivated by the idea that humans prefer these descriptive properties in some order. For instance, humans prefer types of objects (e.g. chair, table, etc), followed by absolute properties like colour (e.g. red, brown, etc), followed by relative properties like size (e.g. small, big, etc) and orientation (e.g. facing front, facing back, etc). The algorithm iterates over the above order to see if adding a property to the description will help disambiguate the target from its distractors and if so, the property is added to the description. The algorithm iterates until a description is found that completely disambiguates the target entity from its distractors or if it fails to find one because it ran out of properties. While properties are added to the description only when they have distinguishing character, the type information is always added to the list. For example, there will never be a referring expression "red". However, the algorithm does not guarantee a minimal expression. For instance, attributes added to the description during earlier iterations cannot be removed when they are found to be redundant later. However, this shortcoming is claimed to be psychologically plausible in the human referring expression generation mechanism. Others have studied relational attributes (Dale and Haddock (1991); Krahmer et al. (2003)), expressions for entity sets (van Deemter (2002); Gatt and van Deemter (2009)), etc using the algorithmic approach. Several algorithms of different computational complexities and efficiency for similar purposes have been proposed (Dale and Haddock (1991); Reiter and Dale (1992, 1995); Krahmer and Theune (2002); van Deemter (2002); Krahmer et al. (2003); Siddharthan and Copestake (2004); Belz and Varges (2007); Gatt and Belz (2008); Gatt and van Deemter (2009)).

2.4.1.2 Planning approaches

Some of the earliest planning approaches to referring expression generation were presented by Cohen (1981); Appelt (1985). Heeman and Hirst (1995) presented a computational framework of how dialogue partners collaborate in generating and clarifying referring expressions. The model proposed two primitive actions: *s-refer* and *s-attrib*, using which referring expressions can be generated. The *s-refer* is an action that is performed by the speaker to convey his intention to generate a referring expression and the *s-attrib* is an action that ascribes some attribute like category, colour or shape to the referent. Koller and Stone (2007) presented a planning approach to sentence generation (described in section 2.3) that was extended to referring expression generation as well. They also adapt to the hearer's knowledge of domain entities by keeping track of what the hearer knows, does not know and the potential confusion the hearer may have among the domain entities. The goal state is defined as a referring expression that does not refer to any distractors or entities that the hearer does not know about. Therefore, the referents that the hearer knows are appropriately introduced (e.g. "the rabbit" instead of "a rabbit"). The hearer's knowledge is modelled using a static user model like most other systems.

2.4.1.3 Statistical approaches

Poesio et al. (1999); Cheng et al. (2001); Stoia et al. (2006); Greenbacker and McCoy (2009) present supervised learning approaches to referring expression generation. Poesio et al. (1999) and Cheng et al. (2001) use a manually annotated corpus of museum descriptions to study and model the production of parts of referring expressions that do not serve to disambiguate the intended referent from distracting entities but to provide additional information about the entity being referred to. These are called *non-referring* parts. They model this using a decision tree learned automatically from the corpus using supervised learning methods.

Stoia et al. (2006) also presented a decision tree learning approach to choose appropriate referring expressions in a virtual world navigation task based on the discourse history, spatial features like the user's view angle, distance from the target, distractors in the view, etc. The algorithm was evaluated by humans who judged the output to be better or as good as human output 62.60% of the time.

Similarly, Greenbacker and McCoy (2009) used supervised learning methods to learn decision trees to choose between different types of *main subject referential expressions* (MSRE). A main subject referential expression is a referring expression that refers to the main subject of the article. They use a human annotated corpus called GREC, which contains Wikipedia articles about countries, rivers, cities and people, and choose between names, pronouns and noun phrases (of varying length) to refer to the main subject referent. Features for learning a decision tree were carefully chosen based on their utility as reported by psycholinguistic studies. Different versions of decision trees trained using different sets of features produced an accuracy between 68.2% and 72.6% when tested on the testing corpus.

2.4.2 Opportunities for adaptation

In many complex domains like technical support, dialogue systems must adapt to users at the level of referring expressions in addition to other dimensions such as complexity of instructions. Previously, some REG modules adapt to users' domain knowledge (Dale and Haddock (1991); Reiter and Dale (1992); Koller and Stone (2007)) and to their physical environment (Stoia et al. (2006)). Most of the above studies in REG adapt to users by consulting a static user model that contains information about what (parts of) referring expressions they already know. In a sense, these approaches assume that a completely trustable user model is available before they output utterances or texts adapted to their users. However, we argue that it is not always the case. In most cases, users using interactive systems are unknown to the system. To be adaptive to such users, these systems have to analyse the user's traits and adapt to them online during the interaction. In this thesis, we attempt to model such dynamic adaptive behaviour in referring expression generation in spoken dialogue systems.

Referring expression generation (REG) is usually seen as a translation problem: translating a semantic representation of a referent into a linguistic expression (Reiter and Dale (2000)). But we view it as a choice problem rather than a translation problem (van Deemter (2009b,a)). Given a semantic representation of the target entity, a REG algorithm must choose from several alternatives an appropriate expression that satisfies all the criteria of the system. We see the problem of choice at three different levels:

1. Choice of RE type: What type of expression to use: proper noun, common noun, pronoun or one-anaphora?

2. Choice of attributes (for common nouns): What attributes to include: category, shape, size, colour, location, functionality, etc?

3. Choice of attribute values: What values should the selected attributes take (e.g. red or crimson, James or James Bond or Mr. Bond)?

Several of these choices can be made based on the user's domain knowledge. For instance, how do we choose between a common noun and a proper noun reference for a domain object? For example, we may have to use "the French president" instead of "Nicholas Sarkozy" because the user does not know him by name. How do we choose the attributes once we have decided to use the common noun reference? For example, we may have to use "the red phone in the corridor" instead of "the emergency phone" because the user does not know that the phone can only be used for emergency calls (i.e. attributes like color and location can be used instead of functionality for the same common noun "phone"). In case of attribute values, we might have to use "the cushioned leather armchair" instead of "the recliner" because the user does not know that the entity also belongs to a category of recliners (note: the attribute here is the category of the entity). Entities may belong to more than one category and they may be related to each other (e.g. hypernym-hyponym relations like recliners, chairs, furniture) or be independent of each other (e.g. man, American, postgraduate, chef, customer, passenger, etc). The choice of categories may be based on several factors. However, the user's knowledge of the category and the target entity's membership in the category must be taken into account. In summary, user-adaptive REG can be seen as a choice (of referring expression) between what is known to the user and what is unknown. Therefore, what we need is a user modelling component that informs the REG algorithm in making appropriate choices to adapt to the user. In addition to the above, another distinction that can be made is how reference subsequent to the initial one be made using referring expressions. In such contexts, referring expressions are usually shortened but such shortening should take into account what the user knows or recently learned during the previous references.

Although, as described above, there is a hierarchy of levels in referring expression generation where choices have to be made with respect to user's domain knowledge, in this thesis, we present a problem of choice between expressions (and not parts of them) that are known and unknown to the user. Also, we do not focus on subsequent references and therefore only use the same referring expression as the initial one during subsequent references. We build a user modelling component that chooses between a set of referring expressions that are known and another set of expressions that are unknown to users.

2.5 Conclusion

In this chapter, we presented several user modelling techniques ranging from simple representational models to advanced predictive models. In contrast to most models developed for information retrieval tasks, we will focus on learning user modelling strategies for dialogue systems that seek information to populate the user model, predict using incomplete user models and adapt to user's domain knowledge effectively using reinforcement learning techniques.

We also presented some spoken dialogue systems and natural language generation systems that adapt to users. These systems, adapted to different user features like expertise, personality, preferences, goals, environmental constraints, and so on. Although some of them focussed on adapting to user's domain knowledge like we do, they do so either using rule based or supervised learning approaches. Also very few systems adapt dynamically which we argue is an important factor in building adaptive interactive systems. In contrast, our thesis focuses on dynamic adaptation to user's domain knowledge in interactive dialogue contexts. We also presented an overview of the state-of-the-art in referring expression generation, opportunities for adaptation in REG and finally, how our work on user modelling fits into the larger framework of REG.

Chapter 3

Reinforcement Learning

3.1 Introduction

Reinforcement Learning is a computational learning framework for stochastic planning in which the learning agent learns the optimal action to take in a non-deterministic environment to maximize its expected long term rewards over a sequence of actions (Kaelbling et al. (1996); Sutton and Barto (1998)). It is a framework where the agent learns from its *trial-and-error* interactions with its environment based on the environment signals. The environment signals include reward and punishment and the objective of the agent is to accumulate rewards and avoid punishments.

It is different from the two most popular machine learning techniques: *supervised* and *unsupervised* learning. Supervised learning is a way of *learning from examples*, in which there is an oracle who tells the agent what to do in different contexts by presenting the agent with example context-action pairs to learn from. In a way, the agent learns from positive and negative examples and generalizes them to new unseen contexts. On the other hand, unsupervised learning presents a way for learning agents to cluster all available contexts into several distinct groups whose members are closer to each other than the members of other groups.

In contrast to these two approaches, in reinforcement learning the agent gets a numerical reward/punishment signal from a stochastic environment depending on the action it takes in different contexts and learns using *trial-and-error learning* techniques. The reward signal suggests to the learning agent how good or bad the action or a sequence of actions have been. Therefore, an advantage of reinforcement learning over supervised learning techniques is that it can be applied to tasks where actions cannot be classified strictly as right or wrong but be rewarded based on how good or bad its effects were to achieve the agent's goals. Another key feature is that, while supervised learning, informs the learning agent if its actions are correct or not after every action, reinforcement learning agents can be rewarded after a sequence of actions, thereby learning an optimal sequence of actions. While using supervised learning methods, an agent learns to imitate the experts and therefore can only learn strategies that are as good as the experts in the corpus, reinforcement learning agents can learn better strategies by searching the policy space to maximize the expected rewards (Rieser (2008)).

3.2 Learning agent and its Environment

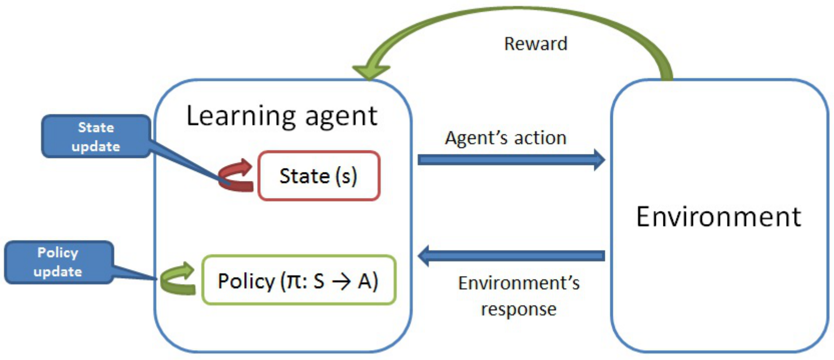


Figure 3.1: Reinforcement Learning agent in environment

Figure 3.1 shows a reinforcement learning agent interacting with and learning from its environment. The agent is situated in an environment with which it interacts. The agent is capable of a set of actions and it senses responses from the environment. The environment also sends a reward signal to the agent. The reward signal is a scalar numeric signal and proportional to how good or bad the agent's actions were in the environment. Often, the signals are positive (i.e. rewards) when the agent's actions are favourable to the environment and negative (i.e. punishment) when they are not. The agent has an internal representation of the environment context as it observes it, which is called the *state* of the agent. At each step in this interaction, the agent's action changes the state of the environment. The agent then updates its state based on its observations of the environment or the environment's responses. The learning task therefore is to choose the optimal action for each of the states that maximises its sum total of the expected reward signals at the end of the interaction (i.e. total reward). In order to do so, the agent should learn to map the most appropriate action that it can take in a given state. This mapping is called the agent's policy (or strategy). Usually, the agent starts with a random policy (i.e. choose actions at random for any given state) and updates its policy based on the rewards from the environment until it figures out the optimal action for each of its states.

An example of a reinforcement learning task would be that of a robot learning to avoid obstacles on its path from point A to point B (fig 3.2). It may have a set of actions it can take to interact with its environment, like moving forwards and backwards, turning left and right, etc. It may have sensors like bump sensors and sonars using which it can observe the state of its environment. Let us assume that its proximity to point B can also be sensed using a light sensor and that point B is a light source. By giving the robot a negative reinforcement (i.e. punishment) whenever it bumps into a wall or other obstacle, the robot can be made to learn to take evasive action in environmental states where the sensors indicate proximity to obstacles in a particular direction. Similarly, by giving a positive reinforcement (i.e. reward) when it reaches point B, it can be made to learn a sequence of actions to follow the light to reach the source. Therefore, through reward and punishment, the robot can be made to learn a policy to navigate from point A to point B avoiding obstacles on the way. It is also possible that there may be several paths (shown by many dotted lines) to take and one path may be better than others in a certain way. For instance, a certain path may be easier to navigate and another faster in terms of time and distance. It is possible to make the robot learn such optimal paths by incorporating these parameters in the reward function by rewarding shorter paths more than longer paths.

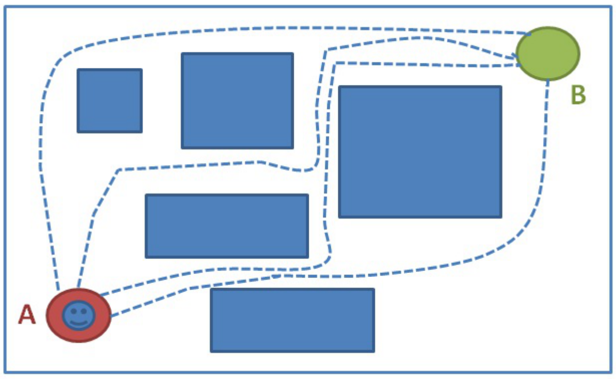


Figure 3.2: Robot path navigation

3.3 Markov Decision Processes

The learning problem of the agent is usually modelled as a Markov Decision Process (MDP) (Bellman (1957)). It is a formal representation of the state space of the environment, the actions that the agent can take, the possible transitions from one state to another by taking various actions, and the reward the agent gets for such transitions. An MDP can represented as a 4-tuple as follows:

$$< S, A, T, R >$$

S - Set of states $(s \in S)$ of the environment

A - Set of actions $(a \in A)$ that the agent can take

$T(s_t, s_{t+1}, a_i)$ - Probaility of transitioning from state s_t to state s_{t+1} by taking action a_i

$R(s_t, s_{t+1}, a_i)$ - Reward signal for transiting from s_t to s_{t+1} by taking action a_i

At each time step t, the agent can traverse through the MDP from one state s_t to another s_{t+1} ($s_t, s_{t+1} \in S$) by taking action a_t ($a \in A$). The transition to the next state s_{t+1}, however, is probabilistically determined by T. During such transitions, it receives a reward r_{t+1} (where $r_{t+1} \in R$). Usually, the learning task is assumed to satisfy the *Markov property*, so that the environment state transitions (to the next state s_{t+1}) are defined based only on the previous state (s_t) and the agent's current action (a_t). Therefore, the probability of transiting from one state (s_t) to the next and the probability of obtaining a reward r_{t+1} at time step t is given by.

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, s_{t-2}, a_{t-2}..s_1, a_1) = P(s_{t+1}|s_t, a_t)$$
$$P(r_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, s_{t-2}, a_{t-2}..s_1, a_1) = P(r_t|s_t, a_t)$$

In a Markov Decision Process (MDP), we make an assumption that the learning agent is able to correctly observe its environment and update its internal state without any uncertainty (i.e. they are fully observable). However, it is not always the case. In the case of our robot example, the robot's sensors may be faulty or other environmental circumstances may not allow the sensors to make accurate measurements of the environment around the robot. In such cases, the internal state of the agent is treated as *belief state*, which represents the agent's beliefs of its environment. Such models are called *Partially Observable Markov Decision Processes (POMDP)* (Kaelbling et al. (1998)). In this thesis, we assume that the user's actions are fully observable and therefore use MDPs to represent the user modelling problem. However, we do not rule out the possibility of improved performance if our problem is modelled as a POMDP. We discuss scenarios where POMDPs can be applied in chapter 9.

3.3.1 Learning objective

The objective of the learning agent can be defined as the total reward that it accumulates over several steps (n) in the future starting from t (denoted by $total\_reward$ below).

$$total\_reward_t = r_{t+1} + r_{t+2} + ...r_{t+n}$$

Reinforcement learning also uses the concept of *delayed rewards* (Watkins (1989); Barto et al. (1990)). Delayed rewards is a concept of rewarding the agent after a sequence of actions that it takes in its environment. By using delayed rewards, the agent can be made to learn a sequence of actions that would lead it from the start state to the goal state where it would be rewarded handsomely. In this thesis, we use delayed rewards in the sense that we reward the agent at the end of the conversation. It makes the agent learn a sequence of decisions to both populate the user model and to use it efficiently to adapt to the user and to trade-off between the two goals. These type of problems are also called *sequential decision making* problems. In such tasks, it is also possible to discount the future rewards using a *discounting factor* (γ). The $total\_reward$ can then be rewritten as follows.

$$total\_reward_t = \sum_{i=0}^{n} \gamma^i * r_{t+i}$$
$$\text{where } 0 \leq \gamma \leq 1$$

The discounting factor determines the value of future expected rewards. When $\gamma = 1$, future rewards are considered as important as the current reward. As it approaches 0, the importance of the future reward is reduced. In such a case, the learning agent is said to be short-sighted as it considers the current reward as the most important. There are tasks where the learning agent has to learn to take less rewarding actions at first to enable it to get far higher rewards later than it could have got if it had taken actions that would have given it the highest reward at every state earlier. Such a trade-off is seen in many tasks. Since the action of the agent affects the state of the environment, sometimes taking less rewarding actions could possibly take the agent through a different path than taking highly rewarding actions and such paths may contain avenues to get even more rewards. Tasks such as these suggest that learning agents must be mindful of future rewards as well and not just current or immediate rewards.

3.4 Policy Learning

3.4.1 Policy

A policy ($\pi : s_i \rightarrow a_j$) maps every state of the agent to an action. Sometimes, a policy is also called *strategy*. The policy that enables the agent to choose optimal actions to get the maximum *total_reward* is called the optimal policy (π^*). In order to choose the optimal actions at each state, the learner must know the expected rewards for each of the actions that it has at its disposal. The expected rewards are defined by a state value function ($V^\pi(s)$) for each state and an action-value function (also called Q-value function, $Q^\pi(s,a)$) for each state-action pair for any policy π. The value function ($V^\pi(s)$) for each state is the expected value of total reward that the agent will get if it takes an action from state s and continues taking actions thereafter following the policy π. Similarly, the Q-value of taking an action a in state s is the expected value of the total reward that the agent will get if it starts from s and takes action a and thereafter takes actions from the next state according to policy π. We use E_π to signify the expected reward.

$$V^\pi(s) = E_\pi\{total\_reward_t | s_t = s\} = E_\pi\{\sum_{k=0}^{n} \gamma^k r_{t+k+1} | s_t = s\}$$

$$Q^\pi(s,a) = E_\pi\{total\_reward_t | s_t = s, a_t = a\}$$

There can be several policies that an agent can follow to reach the goal state. However, the objective of solving the task using reinforcement learning is to find an optimal policy π^*. An optimal policy is better than or equal to all other policies. By following the optimal policy, the agent is guaranteed to achieve the maximum reward in the long run. The optimal policy (or policies) (π^*) have the optimal value and Q-value functions. They can be defined as follows.

$$V^*(s) = max_\pi V^\pi(s)$$

$$Q^*(s,a) = max_\pi Q^\pi(s,a)$$

Finally, an optimal policy is the one that maps every state of the agent to the optimal action that the agent can take in order to obtain maximum reward in the long run. It can be defined as the action that has the largest Q-value in the given state.

$$\pi^*(s) = argmax_a\{Q^{\pi^*}(s,a)\}$$

3.4.2 Exploration vs Exploitation

Reinforcement learning algorithms introduce a trade-off between taking actions that are known to deliver good rewards and taking new actions whose results are unknown. During learning, the agent sometimes takes tried and well-tested actions which it knows would deliver the best expected reward. This process of choosing an action that is known to deliver high reward is called *exploitation*. However, at other times it takes untried suboptimal actions with lower Q-values in order to find out whether they might turn out to return more reward then the current optimal action. This process of choosing new and untried actions is called *exploration*. Exploration allows the agent to learn new actions in certain states and exploitation allows it to use what it learned during exploration. A mixture of exploration and exploitation over a large number of trials allows the agent to learn optimal actions in the different states of the state space that will maximise the overall total reward. This behaviour is controlled by the exploration variable (ε). ε is the probability that the actions taken by the learner are exploratory actions. The value of ε can be gradually reduced over time in order to reduce exploration and increase exploitation of high scoring actions towards the end of the learning phase. This decay is controlled using a halving-time parameter that defines when the ε is to be halved. For example, if the having-time parameter is n cycles, the ε will be halved every n cycles.

3.4.3 Algorithms for learning

State value and action values can be estimated using different classes of reinforcement learning algorithms like *Dynamic Programming (DP)*, *Monte-Carlo methods* and *Temporal difference (TD) methods* (Sutton and Barto (1998)). Dynamic programming methods require a complete and accurate model of the environment. In other words, we need the transition probabilities (T) and rewards (R) in order to calculate the Q-values of all state-action pairs and therefore discover the optimal policy. However, in several tasks, the environment models are not readily available. The agent can only get to know the environment's responses and reward signals by interacting with it (i.e. through experience). Monte-Carlo and Temporal difference methods, in contrast to Dynamic programming methods, do not require a complete specification of the environment model for learning.

Monte-Carlo methods follow a two-step process: *policy evaluation* and *policy improvement*. Policies are evaluated and improved in episodic cycles from the start state to the goal state and therefore these methods can be used only for episodic tasks with small and finite MDPs. Temporal difference combines the ideas of Monte-Carlo and Dynamic programming. Like dynamic programming, TD methods estimate the current estimate of reward from the previous learned estimate. This process is also called *bootstrapping*. And like Monte-Carlo methods, TD methods learn from their experience in the environment and not using a model. Unlike Monte-Carlo methods which revise their policy after every episode, TD methods revise their estimates incrementally after every step.

Since we don't have an accurate model of the environment, we don't use DP algorithms for policy learning. It is not clear which of the other two methods: Monte-Carlo and Temporal Difference, is most appropriate to our user modelling problem. We use a temporal difference learning algorithm called SARSA (Rummery and Niranjan (1994)) to learn the action-value function to estimate the expected rewards for different state-action combinations.

SARSA is a algorithm which starts with an arbitrary policy and optimizes it whilst exploring it. The algorithm for action-value function update is given in Table 3.1. The learning starts with a policy with arbitrary Q-values for state action pairs. Based on action-value function Q and the exploration variable ε, actions are chosen as the learner hops from one state to another. The learning agent receives a reward, every time it takes an action and transits to a new state. The Q-value of the state-action pair is updated based on its previous estimate and the reward and the Q-value of the action it takes at the new state as follows:

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

α is called the *learning rate* ($0 < \alpha < 1$) which determines how fast or slowly the algorithm learns from its experience. The name SARSA comes from the quintuple $< s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1} >$ in the above update equation. In this study, we use a reinforcement learning toolkit called REALL (previously called ICARUS) which implements the SARSA algorithm (Shapiro and Langley (2001, 2002)).

3.4.4 Linear function Approximation

Linear function approximation is a generalisation technique which is used in reinforcement learning algorithms to scale them up to states which were not visited during training. So far, we have assumed that the learned state-values for states or action-values for state-action pairs are stored in the form of a table. However, the number

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

Table 3.1: SARSA algorithm (Sutton and Barto (1998))

of entries in the table could grow exponentially with the task complexity. The problem with huge state spaces is that learning becomes more difficult as it requires more time and data to visit all possible states. Generalisation techniques like linear function approximation have been employed to resolve the problem of large state spaces (Henderson et al. (2008)). This is an instance of *supervised learning* that takes examples from any desired function (e.g. value or action-value functions) and generalizes them to construct an approximation function. Instead of training the agent with a large state space, we could train it with a smaller state space with linear function approximation, which could then be employed to handle large state spaces during evaluation with real users. The generalisation function can therefore produce a good approximation over unseen states.

Let Q' be the linear approximation function for the action-value function Q presented in the SARSA algorithm. Each state (s) is represented as a column vector of state variables ($\phi_s = (\phi_s(1), \phi_s(2), .., \phi_s(n))^T$). Q' is a linear function of a parameter vector ($\theta_a = \theta_a(1), \theta_a(2), .., \theta_a(n)$) for each action ($a$) and the state vector ($\phi_s$) and is given by,

$$Q'(s,a) = \sum_{i=0}^{n} \theta_a(i)\phi_s(i)$$

The learning algorithm learns the parameter vector (θ_a) for each action (a) pair, using which Q-values for any state-action pair can be predicted. The method therefore generalises even to states unseen in the training phase by computing how similar unseen states are to those that were seen during training.

3.5 Applications of Reinforcement Learning techniques

Reinforcement learning methods have been successfully applied to solve many problems. Machine learning methods offer a lot of advantages like data-driven and low-cost development, optimal policies, generalisation to unseen states, etc. Reinforcement learning has been used to solve problems like mobile robot navigation (Malmstrom et al. (1996); Smart and Kaelbling (2002); Su et al. (2004)), air traffic management (Alves et al. (2008)), diagnosis using medical images (Netto et al. (2008)), etc. Reinforcement learning has been also used for adaptive interactive page recommendation (Hernandez et al. (2003, 2004); Taghipour et al. (2007)). Taghipour et al. (2007) presents page recommendation as a Q-learning problem. A window of pages visited by the user and those previously recommended by the system are treated as the state and the choice of pages that the system could recommend is treated as the set of actions.

3.5.1 Dialogue Management Strategies

We discussed dialogue management briefly in section 2.2. In dialogue management, the system has to decide how to act in a dialogue situation which is decided by a dialogue management policy. These policies are sometimes manually coded using finite state machines and standard markup languages like Voice XML, Call Control XML, State Chart XML, etc (Seneff and Polifroni (2000); Pietquin and Dutoit (2003); Brusk et al. (2007); Griol et al. (2010)). The performance of the dialogue management module directly depends on how good the dialogue policy is. The task of designing a good dialogue policy is therefore considered more of an artistic task rather than an engineering task. Such policies are very difficult to be manually coded and maintained by dialogue system designers as the dialogue task becomes more complex thereby increasing the dialogue state space. Machine learning methods like reinforcement learning have been successfully used to learn dialogue policies in increasingly complex and uncertain dialogue scenarios (see below).

Levin et al. (1997) showed how to describe a dialogue system in terms of states, actions and strategies and how to use reinforcement learning to learn such strategies describing what action to take in different dialogue states. The dialogue state ($S_{s,t}$) represents the knowledge of the system at any given point of time about itself (i.e. goals, resources available, filled slot values, etc) and the environment (i.e. the user, noise conditions, database, etc). The dialogue action set ($A_{s,t}$) includes all possible actions that the dialogue manager can take. This includes greeting the user, asking for information, presenting information, confirmation of known information, etc. Sometimes more complex actions are also used like asking for more information while implicitly confirming old information. Finally, a dialogue strategy ($\pi_{dm}(S_{s,t}) \rightarrow A_{s,t}$) is the component that maps the dialogue state to the most optimal action. The above setup is represented as a Markov Decision Process (MDP) and the dialogue strategy is optimised to maximize a long term reward. The reward could be modelled on parameters like dialogue duration or on user satisfaction scores. Therefore, the agent's goal is to learn a dialogue management strategy that maximizes the user's satisfaction score or minimizes dialogue duration and so on.

During the learning phase, the dialogue manager needs to interact with real users. In general, this is very difficult, as the learning phase for any RL agent requires thousands of dialogue interactions with users before it can find an optimal policy. Besides, the system in learning mode would explore different dialogue actions that can be very bizarre and annoying to real human users. Real users can therefore not be obviously used during the learning phase. On the other hand, the learning agent could learn from a dialogue corpus. However, such corpora have to be extremely large to cover all of the dialogue state space and still produce the variety of user behaviour. Since such huge corpora are not always readily available or created, user simulation models were proposed to replace real users (Eckert et al. (1997)). The user simulation models are themselves built using a small dialogue corpus containing real user interactions. These models were further smoothed and discounted, so that they generalise to produce unseen behaviours in the corpus. User simulations can interact with the learning agent indefinitely producing a variety of dialogue behaviour that we need for learning a good policy. We describe user simulation models in detail in chapter 6. Using the above setup (figure 3.3), it has been shown that reasonably good policies could be learned automatically from small amounts of corpus data. Many studies have examined more complex dialogue management problems based on the above model. Schatzmann et al. (2007b) uses this approach to learn DM strategies that handle uncertain noise conditions at the user's end. The system learns when to appropriately produce confirmation moves to ground mutual information about the user's goals. Typical system dialogue actions include asking for slot information, confirming slot information, presenting results, etc. Similarly, Rieser and Lemon (2009a) use this framework to learn to choose between different modalities (speech vs display) to present information in a dual task situation (with high cognitive load on the user) based on user's environmental constraints and the database hits. Henderson et al. (2005, 2008) extended the simulation based RL framework to handle large state spaces using linear function approximation.

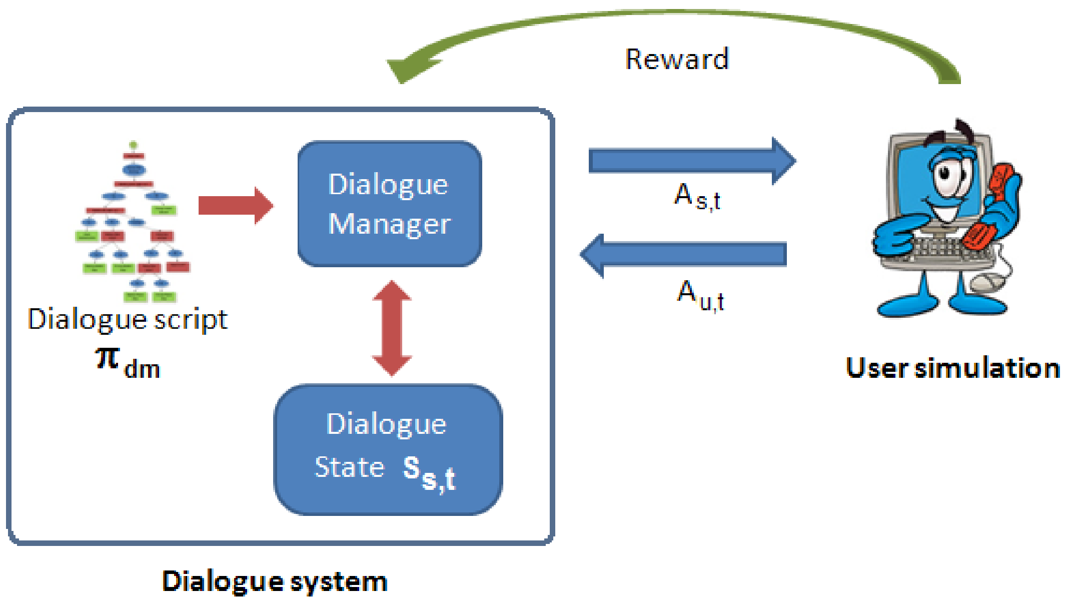


Figure 3.3: Learning Dialogue Management policy using RL

In contrast to *simulation based* reinforcement learning methods, Singh et al. (1999, 2002); Walker et al. (2000); Tetreault and Litman (2006) presented dialogue management strategy learning using a *model based* reinforcement learning framework. In this framework, the strategies are learned from limited real user data using dynamic programming techniques and not from a simulated user. Because of limited data, the system can only learn strategies for states seen in data. Therefore only partial strategies are learned. In Tetreault and Litman (2006), the RL agent learns using a dynamic programming algorithm called *value iteration* to choose between different kinds of questions to ask the student interacting with the system based on state features like learner's frustration, certainty in response, performance so far, etc. In a similar work, Jokinen et al. (2002) used reinforcement learning inspired methods to learn user interaction patterns from a corpus to build different user models at different levels: individual, group and general. These user models were then used by the dialogue system to adapt to the user's dialogue strategy patterns. Recently, the representation of the dialogue states has been made using partially observable MDPs (POMDPs) to learn dialogue management policies under uncertainty arising from noisy speech recognition and stochastic user behaviour (Young (2006); Williams et al. (2006); Williams and Young (2007)).

3.5.2 Language Understanding

Recently, Branavan et al. (2009); Vogel and Jurafsky (2010); Branavan et al. (2010) presented interesting ways to learn to understand natural language commands using reinforcement learning. Branavan et al. (2009) presented a study in which a reinforcement learning agent learns to map natural language instructions in controlled domains to sequences of executable actions. For instance, the agent learns to decode instructions like “Click Start, point to search, and then click for files or folders” and execute them just the way human users reading a Windows Help document would. Therefore, the actions of the learning agent include “left-click”, “double-click”, “type-into”, etc along with parameters. The state of the agent comprises information concerning the GUI objects that the agent can interact with, current instruction, etc. The agent is either immediately rewarded after every step based on whether there is a correspondence between the environment objects at the new state and the words in the following instruction (i.e. environment reward) or task completion calculated using expert annotations. The study found that with a combination of little annotation and the environmental reward, the learning agent was able to map instructions sequences to action sequences much better that other baseline approaches. Branavan et al. (2010) extend this approach to decode high-level instructions, where there is no one-to-one correspondence between the words and the environment objects.

Similarly, Vogel and Jurafsky (2010) present an agent that learns to associate words in the user’s instructions (i.e. directions) to actual physical actions in a map task. The direction following task is modelled as a sequential decision making problem. The action set of the learning agent includes all possible moves it can make from the current location on the map. Each of these actions specify the next landmark and the cardinal direction (i.e. which side to pass the landmark e.g. left, top, etc.). The state of the agent comprises of the instruction it is currently following and its current position on the map. The instruction is represented in terms of features consisting of spatial terms, landmarks on the map, etc. The reward function rewards when the path taken by the agent on the map is close to that of an expert path (i.e. visiting the landmarks in the same order and orientation). Using the SARSA reinforcement learning algorithm, the agent learns to associate various spatial terms and current location information to the most optimal action such that the final reward is maximized.

3.5.3 Natural Language Generation policies

Recently, Lemon (2008); Rieser and Lemon (2009b) extended the above MDP model for dialogue management policy learning to NLG policy learning. It was shown that RL can be used for learning strategies that make high-level NLG decisions like utterance (or content) planning in an information presentation task in the restaurant recommendation domain. The study examines how information can be presented such that various parameters like utterance length, amount of information conveyed and cognitive load are balanced optimally. The NLG module is the learning agent which has to choose between seven complex actions. Each complex action is a sequence of primitive NLG actions: *summarize*, *compare* and *recommend*. It chooses primitive actions one after another incrementally and at each decision point it consults the dialogue state and updates it. It also chooses how many attributes (of the restaurants) need to be discussed in its utterances. Action selection is based on the dialogue state which consists of number of matching database hits, number of sentences generated so far, and the user's response. An internal user simulation model predicts the user's response to the partial utterance generated so far, which helps the system to decide whether to generate more information or stop. The NLG agent learns to choose the optimal sequence of primitive actions along with the number of attributes for each primitive action based on the changing contexts. In comparison to baseline hand-coded policies, it has been shown that the learned policies perform much better in terms of maximizing the overall reward.

In summary, reinforcement learning has been used to learn policies for dialogue management, language understanding and language generation in dialogue systems. In contrast, we employ reinforcement learning techniques to learn user-modelling strategies for referring expression generation. This promises a fully reinforcement learning approach to dialogue system development in future.

3.6 Conclusion

In this chapter, we introduced reinforcement learning techniques that we use in the following chapters to approach the problem of user modelling for adaptive REG. We presented an introduction to Markov Decision Processes and the notion of optimizing long term rewards. We also discussed policy learning in terms of value functions and action-value functions. We briefly discussed various approaches to policy learning before presenting the SARSA algorithm that we will use to learn user modelling policies. We have also introduced other key concepts like the *exploration-exploitation trade-off* and *delayed rewards*.

We discussed some of the applications of reinforcement learning in various domains. We showed how reinforcement learning techniques have been applied to policy learning in dialogue management, language understanding and natural language generatio in spoken dialogue systems in the past.

In chapter 4, we discuss why reinforcement learning is a suitable candidate to solve our user modelling problem, how the problem can be cast as a Markov Decision Process and how reinforcement learning algorithms like SARSA can be used to learn a user modelling (UM) policy that adapts to unknown users by choosing the most appropriate referring expressions based on their domain knowledge levels. We examine the following questions in the following chapters:

1. How to build an RL framework for learning adaptive REG policies? What should the dialogue state look like for adaptive REG action? What actions should we consider for unobtrusive adaptation?

2. Can the user simulation models used for dialogue management policy be used for learning adaptive REG policies as well? If not, how do we enhance them for user modelling policy learning? What should the new user simulation model be capable of? How to build a simulation using limited data

3. Can we learn a user modelling policy for adaptive REG with an RL framework that will adapt online to users with different levels of domain expertise?

4. Will such a learned policy perform well with real users as well?

5. Does adaptation to user's domain knowledge affect other task parameters like time taken, task success, etc?

Chapter 4

Basic Framework

In this chapter, we present a basic reinforcement learning framework for adaptive user modelling in dialogue contexts. First we analyse the nature of the problem at hand (in section 4.1) and describe why Reinforcement Learning is a suitable method to address this problem. In section 4.2, we describe the task domain that is supported by our dialogue system. We then build the basic framework to represent the user modelling problem as a RL problem (in section 4.3). And finally, we present our 5-step approach to develop a data-driven user-adaptive dialogue system (in section 4.4).

4.1 Analysis of the problem

In this section, we analyse the nature of the problem at hand. We believe that there are three important factors in the problem setting that suggest the use of Reinforcement Learning.

4.1.1 Learning from interactions

Adapting to unknown users involves learning about their domain knowledge and deciding which referring expression to use based on an incomplete user model. In supervised learning approaches, such decisions can be learned from experts using a corpus containing interactions between the expert and a variety of users. However, it requires a large corpus of interactions to learn a good adaptive policy, which is not readily available in several domains. Another problem is that it is not easy to create such a corpus as it is difficult to find an expert who is not only highly knowledgeable in the domain but also an expert at conversing with and adapting to different types of users (Hinds

(1999)).

Instead, we propose an approach to learn to choose the most appropriate expressions from the user's (implicit and explicit) feedback. For instance, users might complain when they do not understand new referring expressions, and they might implicitly acknowledge when they know and understand them. These feedback signals can be used when choosing different expressions in different contexts on a trial and error basis to reinforce the best choices and to learn to avoid the bad decisions taken by the system. Therefore, we hypothesize that by interacting with users the system can learn to model and adapt to the user's unknown domain knowledge during the course of the conversation.

4.1.2 Sequential decision problem

A sequential decision problem is one in which a decision (in our case, to use a particular expression) at a time t at state S_t will update the state S_{t+1} and therefore affect the decisions taken at time $t+1$. Choosing referring expressions based on user's domain knowledge is a sequential decision making problem because the use of inappropriate expressions reveals the user's domain knowledge in the form of user's request for clarifications. This allows the system to take corrective action and adapt to the user. Therefore, the decisions taken by the system in a dialogue context are not isolated decisions. Each decision taken by the system results in the update (or not) of the context based on which the next decision is taken. The REG module has to learn when it is a good time in the sequence to seek information to build the user model and when to adapt in order to maximize the overall adaptation to the user.

4.1.3 Stochastic environment

It should be noted that the information given by the users that is used to populate the user models is not always reliable. Although we can reasonably assume that the users are cooperative and ask for clarifications when presented with an unknown referring expression, they may not always do so. For example, they may like to avoid such clarification questions for some reason or they may misinterpret the expression to refer to an incorrect entity. Such incorrect user responses (or lack of responses) lead to incorrect updates in the user model. Although this may happen occasionally, the system must be robust enough to deal with uncertainties arising from the user's actions.

To summarize, the problem of adaptive user modelling in dialogue contexts is a

All Jargon: Please plug one end of the `broadband cable` into the `broadband filter` .
All Descriptive: Please plug one end of the `thin white cable with grey ends` into the `small white box` .
Mixed: Please plug one end of the `thin white cable with grey ends` into the `broadband filter` .

Table 4.1: Referring Expression examples for 2 entities

sequential decision problem in stochastic environments. Adaptive user modelling policies can be learned by interacting with different types of users. Reinforcement Learning is well suited to learn an agent's behaviour based on environmental responses and rewards (Kaelbling et al. (1996); Sutton and Barto (1998)). Several research studies have used RL to solve stochastic sequential decision making problems in the past (Barto et al. (1990); Littman (1996)). For these three reasons, we propose to explore reinforcement learning techniques to address the problem of user modelling for REG for unknown users.

4.2 Problem domain

We chose to study this problem in a spoken dialogue system environment in a technical support domain. The dialogue system interacts with users and helps them with their technical problems such as troubleshooting or setting up broadband Internet connections. In this setup, the system gives step by step instructions to install a broadband Internet connection as shown in figure 4.1. The system should choose appropriate referring expressions to refer to domain entities in a dialogue setting based on its knowledge of the user. The system could choose to use technical expressions (or jargon expressions) or descriptive expressions or even a mix of the two as appropriate (see examples in Table 4.1).

In this task, instead of making REG choices hierarchically (as described in section 2.4.2), we only choose between technical expressions and more general descriptive expressions. However, this choice is made based on the user's ability to identify the

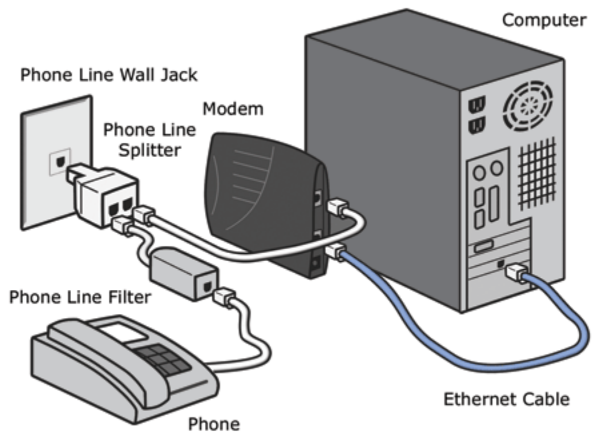


Figure 4.1: Dialogue task domain

domain objects using jargon expressions. Although, this is an important limitation, we make this simplification because our goal is to demonstrate a framework that can be used to learn a user modelling (UM) policy in a dialogue context that chooses referring expressions based on the user's knowledge of the domain.

4.3 Basic framework

As the first step, we model the user modelling problem as a Markov Decision Process (MDP). As explained in chapter 3, an MDP is represented by a 4-tuple $< S, A, T, R >$, where S is the set of states of the learning agent, A is the set of actions it can take to move from one state to another, T is the probability of transition from one state to another and R is the reward the agent gets when it transits between states. We explain how each of these components are modelled to represent the problem of user modelling.

4.3.1 Action set

As discussed earlier, the actions that the agent takes are basically adaptive actions in which it chooses amongst various choices the optimal referring expressions that fit the user's domain knowledge levels. But before that, as a modelling agent, the system should learn more about the partner it is interacting with progressively so that it can adapt better. Hence the action set should reflect the twin objectives of being adaptive as well as sensing its dialogue partner's domain knowledge. One approach would be to design different sets of actions to fulfill these objectives. For instance, we can build explicit questions (e.g. “Do you know what a broadband filter is?”) in order to sense the user’s knowledge of domain objects. However, as discussed earlier, asking explicit information sensing actions are intrusive when used excessively without care and therefore we explore the possibility of using unobtrusive actions for sensing.

In natural human-human conversations, one of the ways the interlocutors acquire information about their dialogue partner’s domain knowledge is based on their partner’s requests for clarification about referring expressions that they cannot interpret. In such cases, the interlocutor’s use of hard-to-interpret referring expressions has two goals: to identify the target domain entity to the dialogue partner and to elicit information on the partner’s domain expertise. This approach is unobtrusive because the user is not being asked to answer questions that relate only to adaptation. We decided to follow a similar *use and clarify* approach where the system learns to use jargon expressions decisively to sense information about the user’s domain knowledge. Here jargon expressions sometimes have two outcomes: sensing and adapting.

We define two unobtrusive adaptive actions. We call them *REG actions* (e.g. *choose-jargon*, *choose-descriptive*, etc). One should note that these two actions represent two different ways of addressing the domain referents. However, the system should also learn to use these actions appropriately based on the user’s knowledge of the domain. These consitute a part of the domain communication knowledge of the system (Rambow (1990); Kittredge et al. (1991)). These actions can be used to not only reflect the system’s adaptive behaviour towards the user, but also serve to elicit information from the users that is required for adaptation in the first place. For instance, the use of unknown jargon would automatically elicit clarifications from users which can then be used to inform the system that the user is unaware of the expression. So by decisively using jargon, the system can gather information to predict the domain knowledge of the user and actively adapt to the user. Therefore, we model our action set to contain only REG actions.

Additionally, we foresee the combined use of jargon and descriptive expressions in order to tutor the users about the domain objects (e.g. “The broadband filter is the small white box with two sockets. Can you plug the broadband filter into the phone socket?”). We call such an REG action *choose-tutorial*. We also add to list of REG actions a choice for the system to use the same expression used by the user. Such a behaviour is called *lexical alignment* or *entrainment* (Pickering and Garrod (2004); Porzel et al. (2006); Buschmeier et al. (2010)). This REG action will enable the system to lexically align with users. Therefore, given the state of the system and the domain

1. choose-jargon
2. choose-descriptive
3. choose-tutorial
4. choose-user-choice

Table 4.2: REG Actions

object to refer to in the current utterance, the system's choice of REG actions are given in table 4.2.

The result of these actions would then be a set of referring expressions $REC_{s,t}$ referring to one or more domain objects mentioned in the utterance. The act of referring expression generation is a part of a conversation between the system and the user. Therefore in addition to the REG actions described above, our framework prescribes the use of a dialogue management module that generates dialogue actions ($A_{s,t}$) that sustains the conversation. In future, this framework can be extended to adaptive user modelling for dialogue management as well. For example, in dialogue management, the system could face adaptive choices such as choosing between simple and complex instructions depending on the user's domain expertise.

4.3.2 State space

The state of the system represents what the system knows about the environment that it acts upon. For a dialogue system interacting with a user, the system state represents all the information it has about the interaction it has with the user and information concerning the user. The state of the system is usually represented in terms of variables. Although there may be variables concerning various aspects of the system, the dialogue, etc, here we focus on those that are used for user modelling. Our objective is to make the system take the actions, shown in 4.2, based on the user's domain knowledge. We therefore represent the user's domain knowledge using a set of variables (a.k.a. *user model*) as a part of the system's state variables. We present a simple vector where each variable records whether the user knows a jargon expression (JE_i) in the domain (see table 4.3).

These variables only represent the system's beliefs about the user's knowledge as they are not pre-loaded into the system from any external source before the interaction starts. It is instead populated dynamically based on interaction and user responses.

$User\_Knows(JE_1)$
$User\_Knows(JE_2)$
...
$User\_Knows(JE_n)$

Table 4.3: User Model

Each of these variables is three-valued. They are initially set as “unknown”, signifying that the system has no idea about the user it is interacting with at the beginning of the conversation. Later these are set to “true” or “false” values based on evidence presented to the system by the user actions.

4.3.3 State transition

Transition from one state to another is based on the action taken by the system and the environmental response to the action. In our context, state transition means the change of system’s beliefs about the user’s domain knowledge. By user’s domain knowledge, we mean the user’s ability to identify referents when they are addressed using technical terms. In order to update the user model, which is the system’s beliefs about the user, the user’s dialogue actions are used. In the training phase, when the system learns how to adapt to users, we use a user simulation model. This is because agent learning requires a lot of interaction between the agent and the users and such interactions would be very exploratory in nature. Therefore, instead of real users with different levels of domain knowledge, we use simulated users during training. However, in the evaluation phase, we use real human users (in chapter 8). The user simulation simulates the user’s dialogue behaviour by responding to the system’s dialogue actions. For instance, it might follow the system’s instructions in a technical domain conversation and acknowledge the instruction (example 1 in table 4.4), provide requested information to the system (example 2 in table 4.4), etc. This list depends on the domain and task handled by the system. However, for adaptive REG, we prescribe that the user simulation model be capable of asking for clarifications on unknown referring expressions (example 3 in table 4.4).

Users’ requests for clarification can be used to signal their ignorance of the expression or the domain concept. Figure 4.2 shows an example of how the user model state can be changed based on the user’s response. Similarly, their acceptance of the jargon

Example 1: System: Please plug the TV cable into the circular socket. User: (carries out the instruction) Yes. I did that.
Example 2: System: Is the power light on? User: No. It is still off.
Example 3: System: Plug the broadband cable into the ADSL socket. User: What do you mean by 'broadband cable'?

Table 4.4: Examples of user responses

can be assumed to indicate their knowledge of the same. Although this is not a safe assumption as users can have misconceptions and this could affect task success, we could reduce the effect of a user's misconceptions by explicitly requesting users before the interaction starts to ask for clarifications if they are not sure of the domain concepts and that misconceptions might leading to delay in solving their problems.

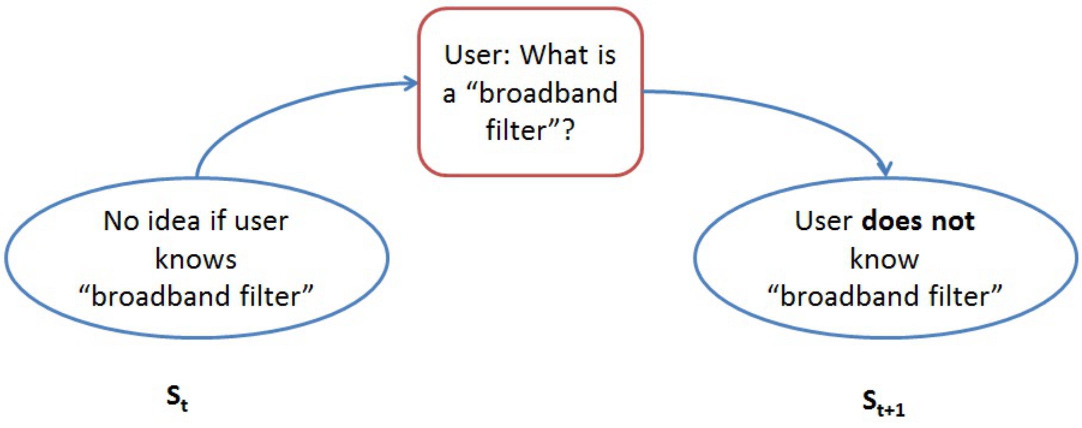

User: What is a "broadband filter"?
No idea if user knows "broadband filter"
User **does not** know "broadband filter"
S_t
S_{t+1}

Figure 4.2: Example of state transition

4.3.4 Reward

Finally, we define the reward function *R* of the MDP. The reward function represents the goals of the system and is a numerical score of the system's performance. We reward the system at the end of each dialogue session for its choice of REG actions.

The system's high-level dialogue management actions are not studied and therefore their choice at different dialogue states are kept deterministic and therefore make no contribution to the reward. But the REG actions like choosing jargon and descriptive expressions at different states affect adaptation. Since our objective is to maximize adaptation, the reward function should be able to present the learning agent with a higher reward it had adapted well to its user and low reward it had not.

Due to the choices of referring expressions made by the system, users might waste dialogue time asking for clarifications, and misinterpret unknown expressions resulting in low task success, increased user satisfaction or learn new jargon expressions and so on. These parameters can be effective indicators of the adaptive behaviour exhibited by the system. For instance, if the system is very adaptive, it might reduce the number of clarifications requested by the user on referring expressions and therefore the number of turns and duration. The overall dialogue time may also be drastically reduced because adaptive behaviour reduces misunderstanding between the users and the system. A reward function could factor in all or some of these parameters to reflect how well the system has adapted. However some questions need to be answered before we decide which parameters to use in the reward function that signifies adaptation. Does dialogue duration and number of turns decrease substantially when the system adapts to a user? Does user satisfaction, user learning or task success increase due to adaptation? Answers to these questions is the key to designing good reward functions.

4.3.5 Learning

RL algorithms are typically used to solve MDPs. The objective of RL algorithms, as described in the previous chapter, is to find an optimal policy that maps each of the above mentioned states of the learning agent to an optimal action. By optimal action, we mean the action that has the potential to fetch maximum long term rewards to the agent. The state of our learning agent represents the system's beliefs about the user's knowledge, and the reward function rewards the agent for choosing referring expressions adaptively towards the user's domain knowledge levels. Therefore, the system must learn to choose the appropriate referring expression for the user in order to receive higher rewards. However it does not initially have any information about the user's knowledge and therefore it cannot adapt. Therefore, it must also learn to acquire information about the user's knowledge before it adapts. Also note that sensing information by using jargon expressions is done at the expense of adapting to the user.

For instance, using a jargon expression may reveal that the user does not know it. But such a move is not adaptive because an unknown expression may be presented to the user. Therefore, the system should know how and when to sense information and when to exploit the sought information to predict the unknown facts in order to adapt to the user. Otherwise, it would simply be sensing more and more information and not be adapting to the user. In case it gets its prediction wrong, it should use that information to adapt later. Therefore, the system has to effectively learn to trade-off between sensing to populate the user model and exploiting the user model to predict and adapt to the unknown user.

During the learning phase, the system is presented with several users with different levels of domain expertise. The system interacts with each user several times over several dialogue sessions. Each user behaves according to its knowledge pattern. For instance, when the system interacts with a novice user and presents a lot of technical expressions, the user requests a lot of clarifications. These requests are assumed to be indications of the user's ignorance and are duly noted in the system's state (i.e. user model). At the end of each dialogue session, the user simulation evaluates the system's adaptation towards the user's domain knowledge and rewards the system proportionally. The reward is higher if the system used the appropriate expressions and lower if otherwise. The system progressively learns to choose an appropriate REG action for the current system state (i.e. user model) such that the final reward is maximized.

Please note that when the system uses jargon expressions that the user does not know, the user could learn them. This means that the user's knowledge is dynamic. However, in this study, we are not modelling this dynamic aspect of the user's knowledge. Our system models only the user's initial knowledge state in its user model. However, in future, it would be interesting to study the learning behaviour of the user as well. Our basic framework prescribes the minimum requirement for presenting the problem of user modelling for REG as a Markov Decision Process. We discuss this further in chapter 7.

4.4 Approach to learning dialogue policies

As discussed in section 3.5.1, dialogue policies have been learned using simulation based reinforcement learning techniques in the past. For this, user simulations were built using dialogues between real users and dialogue systems (Henderson et al. (2005)). However, recently, such dialogues have been collected using Wizard-of-Oz studies and used for building user simulation models (Becker et al. (2006); Schatzmann et al. (2007b)). These practices have evolved into a 5-step process for development of new spoken dialogue systems (Rieser (2008); Rieser and Lemon (2011)). We propose to use this approach to learn and evaluate a user modelling policy in this study.

Step 1. Data collection using Wizard-of-Oz (WoZ) framework: Small amounts of dialogue data are collected using a Wizard-of-Oz framework. In this framework, a human wizard disguises himeself as a dialogue system and interacts with real human users. The interactions are logged and annotated. We describe this in detail in chapter 5.

Step 2. Build User Simulation models using WoZ data: We design a user simulation model that can simulate real user's dialogue behaviour in response to a dialogue system. This model is trained using the data we collected in the previous step. We evaluate the simulation model using appropriate metrics to see how realistic they are in simulating real users. We describe this in detail in chapter 6.

Step 3. Model and train a policy learning agent in a simulated user environment: We model the dialogue system as a reinforcement learning agent in a Markov Decision Process or as a Partially Observable Markov Decision Process. Here, states and action choices of the learning agent are defined. A reward function is designed that models the goals of the agent. We then train the learning agent by letting it interact with the user simulation. The interaction produces thousands of dialogues in which the learning agent first explores different options in various dialogue states and then learns to associate optimal actions with states in order to maximize its long term expected reward. This step is explained in detail in chapter 7.

Step 4. Evaluate the learned policy in user simulation: The learned policy is then evaluated with a slightly varied version of the user simulation model. We compare the performance of the learned policy in terms of objective parameters like reward, number of turns, time taken, etc with baseline policies. This is also explained in chapter 7.

Step 5. Evaluate the learned policy with real users: Finally, the best performing hand-coded policies and learned policies are evaluated with real human users. Users are given tasks or goals and are allowed to interact with dialogue systems implementing the learned and other baseline policies. Like simulated evaluation in the previous step, objective measures are used. Real users are asked to fill out questionnaires that measure user's satisfaction scores. Based on the objective measures and subjective scoring, the policies are compared. We describe this step in chapter 8.

Although, this approach has been used by many for developing and evaluating dialogue management policies Schatzmann et al. (2007b); Rieser (2008) and information presentation strategies Rieser and Lemon (2009b), we adapt it to learn a user modelling policy.

4.5 Conclusion

In this chapter, we have first presented an analysis of the user modelling problem and discussed why reinforcement learning is a suitable candidate to solve this user modelling problem. We have shown how to represent this problem as a Markov Decision Process. We also present an outline of our approach to developing and evaluating learning policies in dialogue systems. In the following chapters, we validate this basic framework using dialogue data from real users and build a data-driven user simulation and learning framework.

Chapter 5

Data Collection

Several studies have used pre-existing corpora of real user's dialogues with dialogue systems to train user simulations (Singh et al. (2002); Henderson et al. (2005); Georgila et al. (2005)). However, such dialogue corpora do not exist in all domains and even if they do, they are not ideally suited for the variety of tasks such as optimizing the system's NLG actions instead of dialogue actions, etc. In such cases, what we need is a dialogue system to collect a dialogue corpus by interacting with real users. This leaves us with a *chicken-and-egg* problem wherein on the one hand we need a dialogue system to build a dialogue corpus with real users and on the other hand we need a dialogue corpus to build a dialogue system. In these circumstances, dialogue system researchers have used the *Wizard-of-Oz* framework wherein data is collected using a "wizarded" dialogue system and such data is later used to build a dialogue system. In this chapter, we use a Wizard-of-Oz (WoZ) framework for data collection in a real situated spoken dialogue task for adaptive referring expression generation (REG). We also present methods to collect additional information like a user's domain knowledge before and after the dialogue task and a user's interaction with his/her physical environment. We later show how such information can be combined to build user simulation models to train adaptive policies for REG (see Chapter 6).

In section 5.1, we present the Wizard-of-Oz framework in general and some previous work. In section 5.2, we describe the WoZ environment designed for collecting dialogues to learn adaptive referring expression generation in detail. Section 5.3 describes the task performed by participants. Section 5.4 describes the data collected in this experiment. Section 5.5 presents an analysis of the corpus data.

5.1 Wizard-of-Oz framework

Wizard-of-Oz frameworks are used in the absence of appropriate dialogue corpora containing dialogues between real users and domain experts or simulated domain experts. It is an effective way to collect dialogues between real users and dialogue systems before actually implementing the dialogue system. In this framework, participants interact with an expert human operator (known as “wizard”) who is disguised as a dialogue system. These dialogue systems are called *wizarded dialogue systems* (Forbes-Riley and Litman (2010)). These wizards replace one or more parts of the dialogue system such as speech recognition, natural language understanding, dialogue management, natural language generation modules and so on. The dialogue partners (the participant and the wizard) are usually seated in different rooms and interact using communication devices like microphone headsets on a computer network. In a setup where the wizard plays the role of a dialogue manager, she hears the user’s responses and usually chooses appropriate dialogue action. However, the wizard’s choices are restricted by the system design and therefore they can only choose those actions that will be available in a fully developed system in the future. Dialogue actions are then converted into natural language utterances and eventually into acoustic outputs using a speech synthesizer. Speech synthesizers are used so that the participants can be made to believe that they are interacting with a spoken dialogue system. The interaction between the system (wizard) and the user is shown in figure 5.1. Wizard-of-Oz (WoZ) frameworks have been used since Fraser and Gilbert (1991) in order to collect human-computer dialogue data to help design dialogue systems. WoZ systems have been used extensively to collect data to learn dialogue management policies (See Hajdinjak and Miheli (2003); Cheng et al. (2004); Strauss et al. (2007)). For example, Whittaker et al. (2002) present a WoZ environment to collect data concerning dialogue strategies for presenting restaurant information to users.

In addition to the dialogue data, users are sometimes requested to fill in questionnaires that ask them to rate the system features on a Likert scale. This information is later used for calculating user satisfaction scores for the reward function using the PARADISE framework (Walker et al. (2000)) based on step-wise linear regression. This framework was considered to be better than relying on intuitions or human-human conversation data to design dialogue system behaviours. Real users interact differently with humans and computers. While their expectations with human interlocutors are high and varied, they are ready to adapt and “go easy” on computers during interaction

(Pearson et al. (2006)). So, in a WoZ framework, the conversation between real users and the wizards are of an appropriate type to be used for dialogue system design. Also, dialogue data collected from human-human conversations are often more difficult to use in dialogue system development because human interlocutors employ a variety of complex strategies to make the conversation successful. It is very difficult to build a dialogue system that is capable of what human interlocutors can do in a conversation. Dialogue systems are usually constrained and are built to be task specific and offer limited freedom to users. Therefore, in order to collect relevant data for development, the Wizard-of-Oz framework is employed.

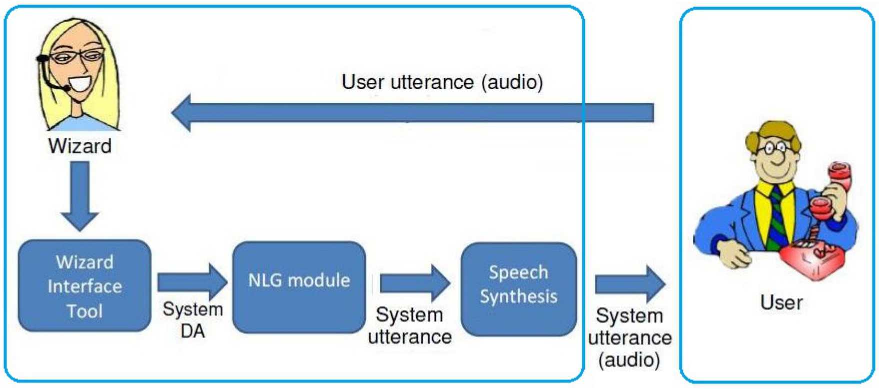


Figure 5.1: Wizard of Oz setup for Dialogue Management

5.2 WoZ for Referring Expression Generation

Our primary objective is to collect user responses to the system's use of different kinds of referring expressions in a technical domain task and to study how participants (hereafter called users) with different domain knowledge and expertise interpret and resolve different types of referring expressions (RE) in a situated dialogue context. We also study the effect of the system's lexical alignment due to priming (Pickering and Garrod (2004)) by the user's choice of REs. The users follow instructions from an implemented dialogue manager and realiser to perform a technical but realistic task – setting up a home Internet connection. The dialogue system's utterances are manipulated to contain different types of REs - descriptive, technical, tutorial or lexically aligned REs, to refer to various domain objects in the task. The users' responses to different REs are then logged and studied. This data will be used to build user simulations to simulate the dialogue behaviour of users with different levels of domain expertise.

However, in our task we do not study dialogue management policies and therefore we do not want the human wizard to make dialogue management choices for the system. Instead we use a dialogue manager with a hand-coded DM policy to manage the conversation. On the other hand, we still need a solution to the problem of speech recognition and annotating the users' utterances into dialogue acts. Forbes-Riley and Litman (2010) presented a "wizarded" dialogue system where a human wizard is employed to listen to the user's utterances and annotate their uncertainty levels for the system to choose an appropriate action. We use a similar setup to collect data to train the user simulation models.

Our framework consists of the Wizard Interaction Tool, the dialogue system and the wizard. The users wear a headset with a microphone and sit in a different room with all domain entities laid out in front of them. Their utterances are relayed to the wizard who then annotates them using the Wizard Interaction Tool (WIT). The WIT interacts with the dialogue manager and sends it an appropriate user dialogue action. The manager responds with a natural language utterance which is automatically converted to speech and is played back to the user and the wizard. The interaction between the dialogue system and a real user is shown in figure 5.2. In contrast to previous WoZ frameworks used for data collection (Whittaker et al. (2002); Hajdinjak and Miheli (2003); Cheng et al. (2004); Strauss et al. (2007)), the human wizard *does not* make strategic decisions on system dialogue actions. The wizard only replaces the speech recognition and decoding modules of a spoken dialogue system. The task of dialogue management is done by the dialogue manager. The wizard stays concealed to the participants. The participants are informed that they will be interacting with a dialogue system and the involvement of the wizard is not informed to them. This is done so that the participants have a reasonable expectation towards the system.

5.2.1 Wizard Interaction Tool (WIT)

We implemented a tool that the wizard can use to interact with the dialogue manager called the Wizard Interaction Tool (WIT) (shown in figure 5.3). It is implemented in the Java programming language. The GUI is divided into several panels.

a. System Response Panel - This panel displays the dialogue system's utterances and RE choices for the domain objects in the utterance. It also displays the strategy adopted by the system currently and a visual indicator of whether the system's utterance is being played to the user.

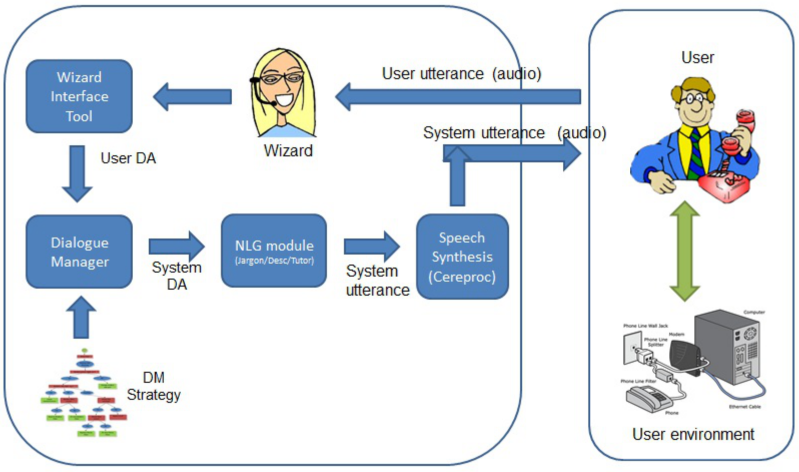


Figure 5.2: Wizard of Oz setup for Referring Expression Generation

b. Confirmation Request Panel - This panel lets the wizard handle issues in communication (for e.g. noise). The wizard can ask the user to repeat, speak louder, confirm his responses, etc using appropriate pre-recorded messages or build his own custom messages.

c. Confirmation Panel - This panel lets the wizard handle confirmation questions from the user. The wizard can choose ‘yes’ or ‘no’ or build a custom message.

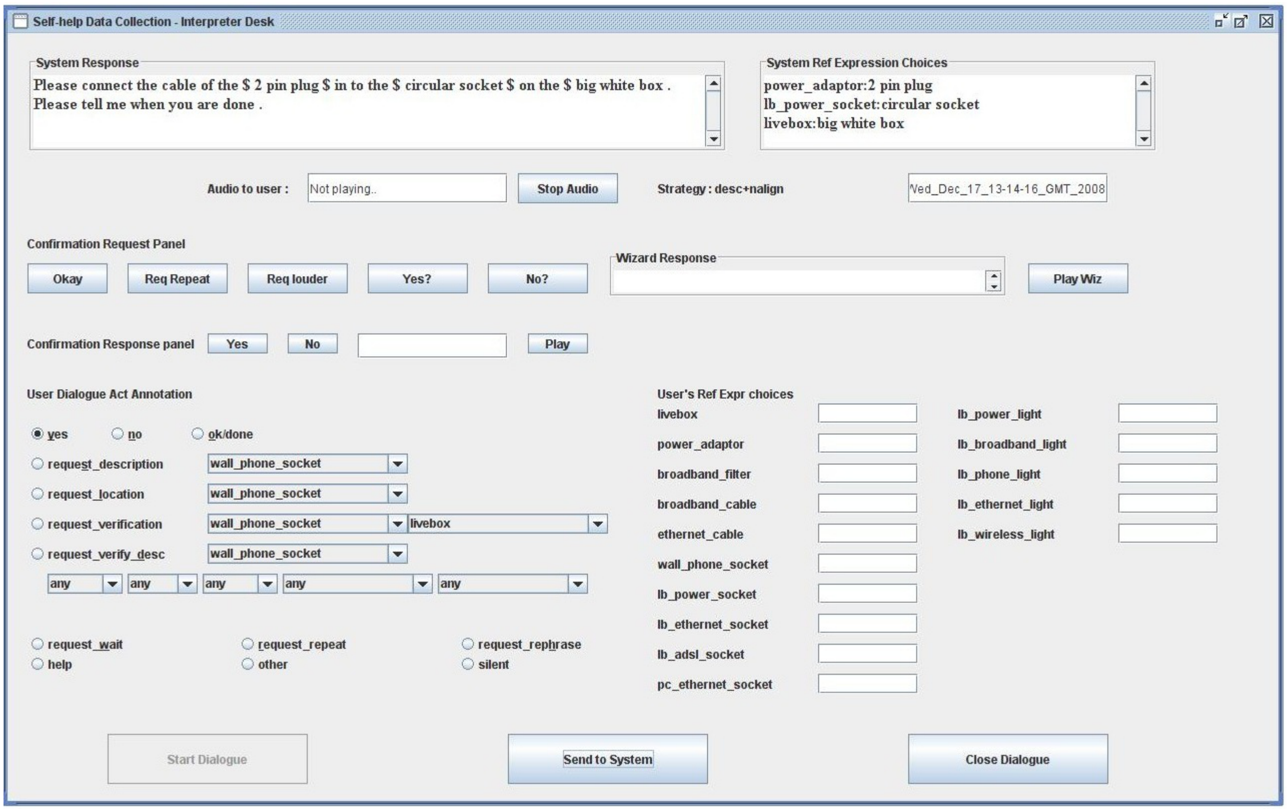
d. Annotation panel - This panel lets the wizard annotate the content of participant utterances. User responses (dialogue acts and example utterances) that can be annotated using this panel are given in Table 5.1. In addition to these, other behaviours, like remaining silent or saying irrelevant things are also accommodated.

e. User’s RE Choice panel - The user’s choice of REs to refer to the domain objects are annotated by the wizard using this panel.

5.2.2 Instructional Dialogue Manager

The dialogue manager was implemented in the Prolog programming language. It drives the conversation by giving instructions to the users. It follows a deterministic dialogue management policy so that we only study variation in the decisions concerning the choice of REs. Our dialogue system has three main responsibilities - choosing the NLG strategies, giving instructions and handling clarification requests. The dialogue system initially randomly chooses the RE and the alignment strategies at the start of the dialogue.

During the conversation, the dialogue manager responds to the user dialogue ac-


Self-help Data Collection - Interpreter Desk

System Response
Please connect the cable of the $ 2 pin plug $ in to the $ circular socket $ on the $ big white box .
Please tell me when you are done .

System Ref Expression Choices
power_adaptor:2 pin plug
lb_power_socket:circular socket
livebox:big white box

Audio to user : Not playing..
Strategy : desc+nalign
Wed_Dec_17_13-14-16_GMT_2008

User Dialogue Act Annotation
[✓ yes] no ok/done
request_description: wall_phone_socket
request_location: wall_phone_socket
request_verification: wall_phone_socket, livebox
request_verify_desc: wall_phone_socket
any any any any any

Figure 5.3: Wizard Interaction Tool - Data Collection

tions with system dialogue actions. It provides step-by-step instructions to set up the broadband connection which are hand-coded as a dialogue script. The script is a simple deterministic finite state automaton, which contains execution instruction acts (e.g. “Plug in the ethernet cable into the ethernet socket in the livebox”) and observation instruction acts (e.g. “Is the ethernet light flashing?”) for the user. Based on the user’s response, the system identifies the next instruction. By using a fixed dialogue management policy and by changing the REs, we only explore users’ reactions to various RE strategies.

Similarly, the dialogue system handles two kinds of clarification requests - open requests and closed requests. With open CRs, users request the system for location of various domain objects (e.g. “where is the ethernet cable?”) or to describe them. With closed CRs, users verify the intended reference, in case of ambiguity (e.g. “Do you mean the thin white cable with grey ends?”, “Is it the broadband filter?”, etc.). The system handles these requests using a knowledge base of the domain objects. Although illustrated with system utterances in Table 5.1, the outputs of the dialogue manager are just dialogue actions.

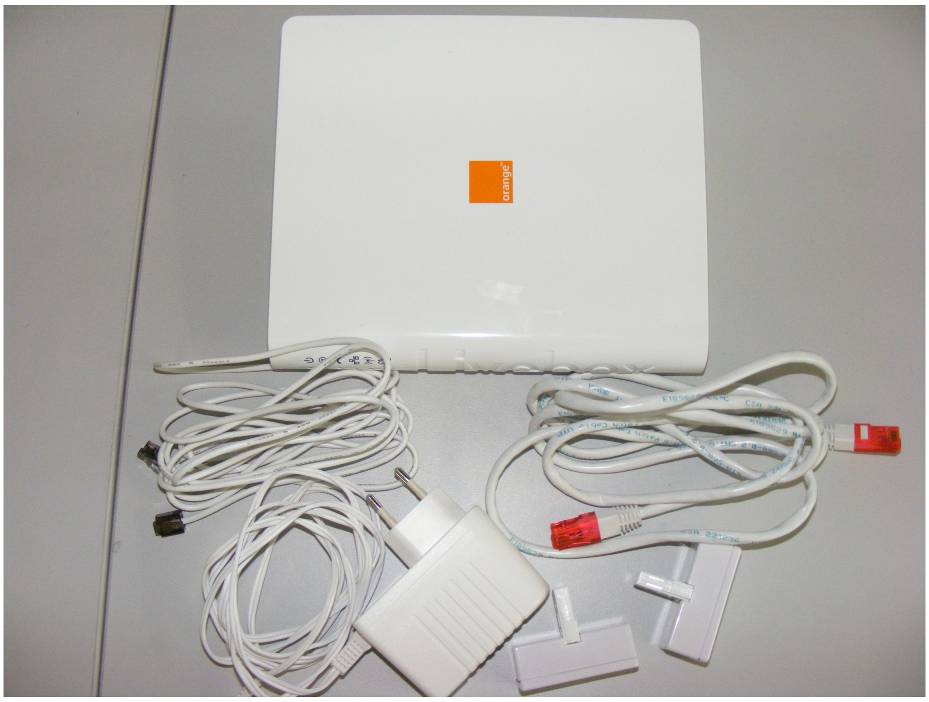


Figure 5.4: Objects in user environment

5.2.3 NLG module

The NLG module produces the system utterances from system dialogue actions and RE and alignment strategies. It produces the system utterances in the Speech Synthesis Markup Language (SSML) format. The NLG realiser uses templates which has references to domain objects as replaceable slots. The following is an example utterance template.

"Plug in the \$ethernet_cable\$ into the \$ethernet_socket\$ in the \$livebox\$"

These slots (e.g. \$livebox\$) are replaced with referring expressions based on the selected strategy to create final utterances. The utterances are finally converted to speech and are played back to the user. We use three strategies for choosing referring expressions:

1. Jargon: Choose technical terms for every reference to all the domain objects. E.g. "Plug in the ethernet cable into the ethernet socket in the livebox".
2. Descriptive: Choose descriptive terms for every reference to all the domain objects. E.g. "Plug in the thick cable with red ends into the square socket with the red stripe in the big white box".

Example Utterance	**Dialogue act**
“Yes it is on”	yes
“No, its not flashing”	no
“Ok. I did that”	ok
“What’s an ethernet cable?”	req_description
“Where is the filter?”	req_location
“Is it the ethernet cable?”	req_verify_jargon
“Is it the white cable?”	req_verify_desc
“Please repeat”	req_repeat
“What do you mean?”	req_rephrase
“Give me a minute?”	req_wait

Table 5.1: User Dialogue Actions with example utterances.

3. Tutorial: Use technical terms, but also augment the description for every reference. Heller et al. (2009) uses a name-then-description strategy similar to that of our tutorial strategy. E.g. “Plug in the ethernet cable in to the ethernet socket in the livebox. The ethernet cable is the thick white cable with red ends. The ethernet socket is the square socket with the red stripe. The livebox is the big white box.”

The above three RE strategies are also augmented with the *alignment strategy*. There are two alignment strategies:

1. Align with user: Use the RE used by the user for referents and ignore the RE strategy.

2. Don’t align with user: Ignore the user’s use of REs and continues to use its own RE strategy.

The NLG module generates system utterances using the algorithm given in table 5.2. It retrieves the list of referents mentioned in the current system dialogue act (using $get\_referents()$ subroutine) and the template for current utterance (using $get\_template()$). It then replaces every referent in the template with appropriate referring expression based on RE and alignment strategies selected by the dialogue manager. The system abandons the existing strategy (Jargon, Descriptive or Tutorial) for a domain object reference when the user uses a different expression from that of the system to refer to the domain object if it is calibrated to align with the user. For instance,

```
Input: System Dialogue Act $A_{s,t}$, RE_strategy $RES$, Alignment_strategy $AS$
Algorithm:
$Ref_{s,t} = get\_referents(A_{s,t})$
$template_{s,t} = get\_template(A_{s,t})$
for each $r \in Ref_{s,t}$
    if $(AS == dont\_align)$ then
        if $(RES == jargon)$ then $replaceall(template_{s,t}, r, jargon(r))$
        if $(RES == descriptive)$ then $replaceall(template_{s,t}, r, descriptive(r))$
        if $(RES == tutorial)$ then $replaceall(template_{s,t}, r, tutorial(r))$
    else
        $replaceall(template_{s,t}, r, user\_choice(r))$
return $template_{s,t}$
```

Table 5.2: WoZ NLG Algorithm

under the Descriptive strategy, the ethernet cable is referred to as “the thick cable with red ends”. But if the user refers to it as “ethernet cable”, then the system uses “ethernet cable” in subsequent turns instead of the descriptive expression.

5.2.4 Speech synthesiser

The utterance produced by the NLG module is marked up with Speech Synthesis Markup Language (SSML), which is input into the speech synthesiser. These are converted automatically into speech by the Cereproc Speech Synthesiser and played back to the user.

5.2.5 The Wizard

The primary responsibility of the wizard is to understand the participant’s utterance and annotate it as one of the dialogue acts in the Annotation panel, and send the dialogue act to the dialogue system for response. In addition to the primary responsibility, the wizard also requests confirmation from the user (if needed) and also responds to confirmation requests from the user. The wizard also observes the user’s usage of novel REs and records them in the User’s RE Choice panel. As mentioned earlier, our wizard neither decides on which strategy to use to choose REs nor chooses the next task instruction to give the user.

5.3 The Domain Task

The domain task for each user was to listen to and follow the instructions from the WoZ system and set up their home broadband Internet connection. See appendix A for instructions given by the system. We provided the users with a home-like environment with a desktop computer, phone socket and a Livebox package from Orange containing cables and components such as the modem, broadband filters, a power adaptor, etc. Figure 5.4 shows a part of the environment setup that was presented to the user. During the experiment, they attempted to set up the Internet connection by connecting these components to each other.

Prior to the task, the users were informed that they were interacting with a spoken dialogue system that will give them instructions to set up the connection. The users were requested to have a conversation as if they were talking to a human operator, asking for clarifications if they were confused or failed to understand the system's utterances. They were also told that misunderstanding might affect task success. The user followed the instructions and assembled the components.

5.4 Data collection

We followed a step-by-step process to collect data from the users. This process not only collected the dialogue exchanges between the user and the system but also collected other information such as the user's domain knowledge before and after the dialogue task, user's interaction with the physical environment and user's review of the dialogue system. We used all this information to build user simulation models and some reward functions (See chapter 6).

Step 1. Background of the user - The user was asked to fill in a pre-task background questionnaire containing queries on their experience with computers, Internet and dialogue systems. (See appendix B.2)

Step 2. Knowledge pre-test - Each user's initial domain knowledge was recorded by asking them to point to the domain object that was called out by the experimenter by its jargon expression. (See appendix B.4)

Step 3. Dialogue - The conversations between the user and the system were logged as an XML file. The log contains system and user dialogue acts, times of system utterances, system's choice of REs and its utterance at every turn. It also contains the dialogue start time, total time elapsed, total number of turns, number of words in system utterances, number of clarification requests, number of technical, descriptive and tutorial REs and number of confirmations. The user's utterances were recorded in WAV format in order to build acoustic and language models for automatic speech recognition in future.

Step 4. Knowledge gain post-test - Each users' knowledge gain during the dialogue task was measured by asking them to redo the pointing task. The experimenter read out the jargon expression aloud and asked the users to point to the domain entity referred to. (See appendix B.4)

Step 5. Percentage of task completion - The experimenter examined the final set up on the user's table to determine the percentage of task success using a form containing declarative statements describing the ideal broadband set up (for e.g. "the broadband filter is plugged in to the phone socket on the wall"). The experimenter awards one point to every statement that is true of the user's broadband set up. (See appendix B.5)

Step 6. User satisfaction questionnaire - The user was requested to fill in a post-task questionnaire containing queries on the performance of the system during the task. Each question was answered in a four point Likert scale on how strongly the user agreed or disagreed with the given statement. Statements like, "Conversation with the system was easy", "I would use such a system in future", were judged by the user and these will be used to build reward functions for reinforcement learning of REG strategies. (See appendix B.3)

5.5 Corpus Analysis

The corpus consists of 17 dialogues from users with different levels of domain knowledge. The participants were from various backgrounds. Some were students and some were professionals. They had different backgrounds from arts, humanities, science, medicine, etc. Each participant was paid 10 after the experiment was finished. They listened to the instructions from the system and carried them out using the domain objects laid in front of them. The experiments examined the effect of using three types of

Parameters	Jargon	Descriptive	Tutorial
No. dialogues	6	6	5
Task Completion rate (%)	98.3	98.3	94.0
Pre-task score (max 13)	6.67	8.5	7.6
Post-task score (max 13)	12.33	10.66	12.2
Turns	28.17	25.83	25.2
CR	3.17	0	0
Sys Words	470.5	471.67	945.6
Time (min)	7.7	6.86	11.72
Time per turn (sec)	16.49	15.9	27.9
URT (sec)	7.47	5.3	5.85

Table 5.3: Corpus statistics (grouped on strategy)

referring expressions (jargon, descriptive, and tutorial), on the users.

5.5.1 Statistics

Out of the 17 dialogues, 6 used the Jargon strategy, 6 used the Descriptive strategy, and 5 used the Tutorial strategy. The task had reference to 13 domain entities, mentioned repeatedly in the dialogue. In total, there are 203 jargon, 202 descriptive and 167 tutorial referring expressions. As expected, users who weren't acquainted with the some domain objects requested clarification on the jargon referring expressions used. More statistics are shown in table 5.3.

Analysis shows that the Jargon and Tutorial strategies produce large learning gain (i.e. difference between post and pre task scores). However, this is not surprising considering the fact that the Jargon and Tutorial are the only strategies that use technical terms for reference. There was an average of 3.17 clarification requests in the Jargon strategy dialogues, whereas there were none in the Descriptive and Tutorial strategy dialogues. This shows that participants of all expertise levels do not question the use of descriptive or tutorial expressions even when they are not appropriate to their domain knowledge levels.

Regarding time, the Jargon and Descriptive strategies produce much shorter dialogues than the Tutorial strategy. This is due to the fact that utterances with jargon and descriptive expressions are much shorter than the Tutorial strategy dialogues (see Sys Words in table 5.3). The Descriptive strategy dialogues are even shorter than the Jargon dialogues because of the absence of clarification requests. URT (user response time) is the time taken by the user to respond to the system's instructions. It also includes the time taken by the wizard to annotate the user's response. This is high in case of the Jargon strategy dialogues. This could be due to two reasons: 1) users may not know the referents and therefore take time to identify them or/and 2) the wizard takes more time to annotate clarification requests. However, the important fact to note here is that, after we subtract URT from Time per turn, we get the time taken to produce the system utterances, which is lowest for jargon utterances owing to the short length of jargon expressions. Based on these factors, we argue that, given the appropriate referring expression based on expertise levels (so that no clarification requests are produced), utterances containing jargon expressions will produce comparable or even shorter dialogues. However, since the current dialogue task is short and not complex enough, there seems to be no difference between the Jargon and Descriptive strategies.

All three strategies produced almost the same level of task completion rates. The mean task completion rate of the Tutorial strategy was a little lower than the other two. However, the difference was not statistically significant. From the above analysis, we conclude that each of the strategies have their own pros and cons. The Jargon strategy may produce shorter dialogues with expert users but with novice users dialogues may be quite long due to clarification requests. On the other hand, they produce high learning gain for novice users. The Tutorial strategy produce high learning gain but at the cost of time. The Descriptive strategy produces shorter dialogues than all other strategies and little learning gain.

We analysed the user scores on the dimension of domain expertise as well. We divided the users into two groups (Group 1 and Group 2) based on their pre-task scores. The mean pre-task score was 7.58. Group 1 consisted of users who scored below the mean (novices and intermediates) and Group 2 consisted of user who scored above the mean (intermediates and experts). The mean pre-task score of Group 1 users was 5.25 whereas that of Group 2 users was 9.66. Group 2 users asked fewer clarification requests (mean = 0.44) than Group 1 users (mean = 1.87). The mean time taken for Group 1 users to finish the task was 9.37 minutes where as that of Group 2 users was 7.88 minutes. The task completion rates (mean) of Group 2 users were a little lower than the Group 1 users. However, the difference was not statistically significant.

Parameters	**Group1**	**Group2**
No. dialogues	8	9
Task Completion rate (%)	97.5	96.6
Pre-task score (max 13)	5.25	9.66
Post-task score (max 13)	11.37	12
Turns	27.5	25.55
CR	1.87	0.44
Sys Words	660.25	566.55
Time (min)	9.37	7.88
Time per turn (sec)	20.94	18.51
URT (sec)	6.63	5.88

Table 5.4: Corpus statistics (grouped on expertise)

Dimension	**Jargon**	**Descriptive**	**Tutorial**
Confidence on task success	3.33	3.16	3.4
Quality of voice	3.17	3	3.2
Easy to identify domain objects	3.16	3.16	3.4
Learned useful new expressions	3	2	2.8
Instructions with right level of complexity	3	3.33	2.2
Conversation of right length	3.17	3.17	2.2
Ease of conversation	3.5	3.3	3.4
Future use	3.33	3.16	2.6

Table 5.5: User scores (mean)

5.5.2 User scores

User scores are subjective ratings given by the users after they finished the dialogue task based on their experience. Users rated the system along several dimensions on a Likert scale between 1 and 4: 1 - strongly disagree, 2 - disagree, 3 - agree, 4 - strongly agree. Table 5.5 presents the mean user scores for the three different strategies.

Firstly, on all three strategies, the users were confident on task success and agreed on good voice quality of the TTS. Surprisingly, users agreed that the domain objects were easy to identify on all three strategies. We suspect that this is so with the Jargon strategy because users were given clarification when they requested for it. On dimen-

Dimension	Group 1	Group 2
Confidence on task success	3.13	3.44
Quality of voice	3.0	3.22
Easy to identify domain objects	3.0	3.44
Learned useful new expressions	2.88	2.33
Instructions with right level of complexity	2.75	3.0
Conversation of right length	2.87	2.88
Ease of conversation	3.38	3.44
Future use	3.12	3.0

Table 5.6: (Mean) User scores based on expertise levels

sions concerning time, the Tutorial strategy was scored the lowest. This acknowledges the fact that the Tutorial strategy utterances were longer compared to the other two. However, the Jargon and Descriptive strategies were not scored differently based on time. In terms of learning new useful expressions, both the Jargon and Tutorial strategies were scored highly as expected. Users scored the Jargon and Descriptive strategies well on the complexity level of the instructions. However the Tutorial strategy was scored lower in comparison. We hypothesize that it may be due to the length of the tutorial expressions.

On two other dimensions of ease of conversation and future use, both the Jargon and the Descriptive strategies were scored almost equally well. We hypothesize that users ignored the fact that the Jargon strategy uses referring expressions that they don't know, because eventually they get a clarification when requested. This seems to impress them rather than frustrate them. Also, the times taken for these two strategies were not very different from each other either. Therefore they were scored almost equally. Although users agreed that the Tutorial strategy conversations were easy, they did not agree to use it in the future. This is because of its information complexity and longer dialogue time.

Table 5.6 shows scores given by users grouped based on their expertise levels. Users of both groups were confident on task success. Users of group 2 found it easy to recognise domain objects more easily and more or less agreed on the complexity of instructions. However, users of both groups found that the conversations were easy and that they would use the system in future.

5.5.3 Shortcomings of the corpus

After analysing the collected corpus, we found that the following features of the corpus were not optimal for the task at hand.

1. **Lack of data to study Lexical alignment:** User utterances were usually very short in our data. Since the dialogue task is a simple instruction giving-following task, each user's utterances are mostly short indicating their acknowledgement to instructions, providing answers to observation questions, etc. When users cannot resolve a referring expression, they request a clarification. The users' utterances didn't have content words referring to domain entities in them. Therefore, lexical alignment between the users and the system could not be studied.

2. **Lack of information on user's domain knowledge**: Another important disadvantage due to shorter user utterances is that we could not estimate the user's domain knowledge from their content. Contentful utterances can be used to learn about the user's knowledge levels based on the concepts they refer to in their utterances. Therefore the only way of knowing their domain knowledge levels is based on their responses like clarification requests, observations and acknowledgements.

3. **Easy task:** The domain task of setting up a broadband connection seemed easy for users. The mean task completion rate of different users was 97.05% (17 dialogues). This can attributed to the fact that we had no real internet connection to test with and therefore had to contend only with the physical tasks of setting up the domain objects in the right pattern and not worry about the on-computer internet settings task.

4. **Labels and contextual cues:** In order to keep the task realistic, we presented the users with a real broadband setup from Orange. Some of the entities that are referred to in the dialogue are clearly labelled and some were made resolvable using contextual cues. Therefore, many users did not have any difficulty in resolving the referents when presented with jargon technical expressions. For example, the ADSL socket was labelled as "ADSL". However there were some entities which were difficult to identify without prior domain knowledge about them (e.g. broadband cable, ethernet light, etc). This resulted in fewer clarification requests than we had expected.

5. **Task repetition:** The broadband installation task cannot be repeated with different strategies with the same user. Therefore, we cannot compare the performance of a user on different strategies. Ideally, the task would be repeatable with different strategies and the user's performances must be comparable against each other.

However, inspite of these shortcomings, the data was still useful in training a user simulation model to learn user modelling policies for adaptive REG, as we show in the following chapters. We were able to build useful user simulation models and train them so that adaptive policies can be learned. We compensated for the deficiency in our data with some domain expertise in classifying the referents and correcting the user knowledge profiles. Therefore, even though the data that we collected were deficient, we did not need collect more data.

5.6 Conclusion

We have presented a Wizard-of-Oz environment to collect spoken dialogue data in a real situated task environment to study user reactions to a variety of REG strategies, including the system's lexical alignment. We have also presented an analysis of the dialogue data and users' feedback on the conversation. The data will be used for training user simulations for reinforcement learning of user modelling strategies for adaptive REG to choose between jargon and descriptive expressions based on a user's expertise in the task domain. In the next chapter, we show how this data has been used to build user simulation models for training and testing user modelling policies for adaptive referring expression generation.

Chapter 6

Statistical User Simulation for NLG

In this chapter, we present new statistical data-driven user simulation models for learning adaptive referring expression generation (REG) policies. They serve as the environment that responds to the learning agent's actions and rewards it for taking different actions at different states. In a dialogue situation, the role of an environment is fulfilled by a dialogue partner. User simulation models replace human users and simulate their dialogue behaviour. They have been used as dialogue partners for learning and evaluating dialogue management policies using reinforcement learning in many studies. They have been used to substitute for real users since using real users for training can be very expensive and they might get frustrated with anomalous (exploratory) behaviour of the system that is still in the learning stage. Even for evaluation, user simulations are used first in order to help iterative development before finally testing with real users.

In this chapter, we explore the following questions: Can user simulation techniques used for learning dialogue management policies be used for learning user modelling policies for adaptive REG? If not, what are the requirements for a new design of user simulation models? How can they be implemented and trained using the limited data that we collected? How well do such models simulate real user dialogue behaviour?

In section 6.1, we present some related work in the domain of user simulation in spoken dialogue systems. Then, we list down the requirements for designing a new simulation model in section 6.2. Section 6.3 describes in detail the novel simulation models. In section 6.4, we present the baseline models whose performance we compare our three-step models to. In section 6.5, we present the smoothing technique used to smooth all the models to handle the problem of data sparsity. Finally, in section 6.6 and 6.7, we compare our three-step models to the other baseline models to show how closely our models simulate real users' dialogue behaviour.

6.1 Related work

Several user simulation models have been used for training the different modules that make up the dialogue system. At the end of training, the system learns dialogue policies that it can use to effectively communicate with users (both simulated and real). Usually, studies that use user simulation models for training have also used them for evaluating the system before final evaluation with real users (Eckert et al. (1997); Scheffler and Young (2001); Chung (2004); Schatzmann et al. (2005); Cuayahuitl et al. (2005); Georgila et al. (2005); Rieser and Lemon (2006); Georgila et al. (2006); Pietquin and Dutoit (2006); Ai and Litman (2007); Schatzmann et al. (2006, 2007b); Ai and Litman (2007, 2009)).

However, some studies have used user simulation models just for evaluation and not for development (Butenkov (2009); Araki and Doshita (1997); Lopez-Cozar et al. (2003)). In order to evaluate a spoken dialogue system, Araki and Doshita (1997) proposed a user simulation so that the dialogue system can be evaluated holistically and objectively for its overall performance as against the methods evaluating the system's modules independently. This in essence captures how the modules of the system cooperate with each other to produce an effective dialogue. In their setup, the dialogue system interacts with the user simulation using text representation of the utterances. The text being exchanged is passed through a coordinating program which introduce linguistic noise to simulate speech recognition errors. The conversations are logged and examined. Similarly, Lopez-Cozar et al. (2003) introduced a user simulation that interacted at the level of acoustic speech signals instead of text. The speech format of user utterances was sampled from a corpus of dialogues.

The interaction between the system and the user simulation models happen in different modes: speech, text, dialogue actions and so on. Chung (2004) presented a user simulation model that interacts with the dialogue system using either text or speech in order to train the speech recognition and understanding modules. The speech format of the utterances was produced using a speech synthesizer. Several user simulation models have been proposed for dialogue management policy learning that interact with the system using dialogue acts. (Eckert et al. (1997); Scheffler and Young (2001); Schatzmann et al. (2005); Cuayahuitl et al. (2005); Georgila et al. (2005); Rieser and Lemon (2006); Georgila et al. (2006); Pietquin and Dutoit (2006); Ai and Litman (2007); Schatzmann et al. (2006, 2007b); Ai and Litman (2007, 2009)).

We can also classify user simulations based on how they are created: *rule-based* and *corpus-based*. In a rule based simulation, the user's dialogue behaviour is defined using hand-coded rules by the dialogue system designers (Guinn (1998); Ishizaki et al. (1999); Lin and Lee (2001); Smith (1998); Chung (2004); Lopez-Cozar et al. (2003)). On the other hand, in the corpus based models, the user's dialogue behaviour is defined by probabilistic models trained on annotated dialogue corpus. Corpus based models capture the uncertainty in the user's dialogue behaviour while the rule based models enforce strictly deterministic behaviour. Stochastic simulation models were therefore employed instead which captures the less likely responses like barge-ins, hang ups, digression, changing goals, etc. Parameters in stochastic simulations can also be set by hand. But such an exercise requires careful consideration of real user behaviour and parameters are usually hard to estimate without looking at data. Another disadvantage is that the parameters set by the designer could be subjective and biased. However, parameters can be estimated objectively and directly using corpus data. Rather than relying on "common-sense" heuristics, parameters can be trained from a corpus containing dialogues collected from real users. The following are some of the corpus-based user simulation models that have been used in reinforcement learning of dialogue management policies.

N-gram models: A bigram user simulation model for training dialogue management policies was first proposed by Eckert et al. (1997); Levin et al. (2000). In this model, the user's dialogue act is decided by the immediate previous system dialogue act as follows.

$$P(A_{u,t}|A_{s,t})$$

This simple model enjoys the advantage of being domain independent. However, it does not produce a realistic behaviour consistent with the user's goals, preferences or even domain knowledge. Georgila et al. (2005) extended the bigram model to n-gram model (n $>$ 2) in order to take some dialogue history into account and therefore produce consistent behaviour.

Goal-directed models: Scheffler and Young (2001) proposed a goal directed model. First, all the possible user moves at different choice points on the dialogue path are mapped out in advance. The choice points are classified as deterministic where the user moves are defined based only on their goals and as probabilistic where the user's responses are uncertain. By identifying probabilistic and deterministic choice points, the user behaviour is restricted to be goal directed. However this approach suffers from being very domain specific and dependent on expert knowledge to map out the all dialogue paths in advance.

Pietquin and Dutoit (2006) proposed a dynamic Bayesian Network model in which the user's dialogue action is conditioned on the user goal and memory as shown below.

$$P(A_{u,t}|A_{s,t}, goal, memory)$$

The goal attributes are ranked based on user preferences which determines how likely is it for the user to drop a certain constraint. It also keeps track of the number of times each constraint is mentioned during the dialogue. The model parameters were initially hand-crafted. This model can be extended to include other user parameters like cooperativeness, degree of initiative, etc.

Linear model: Georgila et al. (2005) proposed a linear feature combination model that calculates the probability of different user actions given the user state using a linear combination of weights for each action and the state features. The weight vector for each action was trained on the user state-action pairs observed in the training corpus.

Cluster models: Rieser and Lemon (2006) proposed a cluster model in which the user state spaces seen in the corpus are clustered and the user responses are modelled based on clusters rather than individual states. During training, the current user state is classified into one of the available clusters and a response is generated. The cluster based model produces responses that are complete (all possible real user actions in a given state are produced) and consistent (no unrealistic user actions in the current state are permitted). Similarly, Ai and Litman (2007, 2009) presented a knowledge consistent user model that behaves like a student responding to a tutorial dialogue system. The student responses are conditioned on the knowledge cluster to which the system's question belongs and the correctness of the student's previous response to the same cluster. This model simulates the learning behaviour of the student. Also, the high learners and low learners are modelled separately.

Agenda-based model: Schatzmann et al. (2007a,c) proposed an agenda based user simulation model in which the user's goals and constraints are stored in the form of *inform* and *request* acts respectively in a stack called the *agenda*. The user's dialogue action is produced therefore by popping n items out of the stack (n is defined as the level of initiative the user takes). The agenda is then updated based on the system dialogue action using the agenda and the goal update models that are estimated from data. The use of a dynamically updated agenda ensures that the user's dialogue behaviour is consistent with the goal of the user.

Although the above models have been used successfully to learn dialogue management policies, they cannot be directly used for user modelling policy learning because they are not sensitive to the system's choice of referring expressions. Ideally, the user simulation must produce clarification requests when the user does not understand the referring expressions used by a system. Also, to learn an adaptive policy, the user simulations need to simulate user groups with different knowledge levels, a feature which the current models do not support.

6.2 Requirements

As presented in section 6.1, several user simulation models have been presented earlier for learning dialogue management policies using the reinforcement learning framework. However, it is clear that those models cannot be used for learning user modelling policies for adaptive REG because such learning requires user simulations to be sensitive to the referring expressions used by the dialogue system. The earlier models respond to the dialogue act (at the dialogue management level) and do not take into account the words or phrases used by the system to address the user. However, the system cannot learn to use appropriate referring expressions unless its dialogue partner is sensitive and responds to their use. For instance, the user simulation should request clarification when the system uses expressions that the user does not know.

Another issue with the earlier models is that they do not simulate a population of users with different levels of domain knowledge. Our objective is to learn a user modelling policy that adapts to users with different levels of domain knowledge. By domain knowledge, we mean the capability of users to identify the referents when the system uses jargon expressions. This is referred to as domain communication knowledge by some researchers (Rambow (1990); Kittredge et al. (1991)). It would not be possible to learn such a policy unless the user simulation can simulate the whole spectrum of users from novices to experts and several intermediates in between by taking into account the knowledge patterns of different users. Finally, like real users, it should simulate learning new technical expressions during the course of the dialogue and has to be consistent with the users' dynamically changing domain knowledge during the conversation. For instance, the likelihood of requesting clarification over a jargon expression that has been previously clarified must be less.

In addition to the above, we require a user simulation model that can be trained using the non-adaptive dialogues that we described in Chapter 5. In summary, our requirements for user simulation are three-fold.

1. Sensitive: Be sensitive to referring expressions used by the dialogue system.
2. Diverse: Simulate a diverse population of users with different domain knowledge levels.
3. Consistent: Learning new referring expressions and be consistent with the (dynamically changing) domain knowledge (i.e. knowledge consistent).
4. Trainable: Be trainable using non-adaptive dialogues.

6.3 User simulation model

In this section, we present data-driven user simulation models which are extended versions of the model presented in chapter 4. The user simulation model consists of the three modules: *the action selection module*, *the knowledge profiles* and *knowledge update module*. All these modules are data-driven and their parameters are populated using the data collected using our data collection framework. Figure 6.1 shows how the collected data drives the different modules of our user simulation model.

6.3.1 Dialogue Action Selection Module

The user's *dialogue action selection* module selects a dialogue action in response to the system's input. Like the other models presented in section 6.1, this model also takes as input the system's dialogue act ($A_{s,t}$). However, with just the system dialogue action, it is impossible to respond in way that is sensitive to the referring expression used. Therefore, we also provide the user simulation with the system's choice of referring expressions ($REC_{s,t}$). This can be seen as a way of abstracting the meaning and the words used the system's utterance. It outputs the user's dialogue action ($A_{u,t}$) and environment action ($EA_{u,t}$). Please note that u denotes user, s denotes system and t denotes time step. This model can be designed to satisfy the first requirement listed

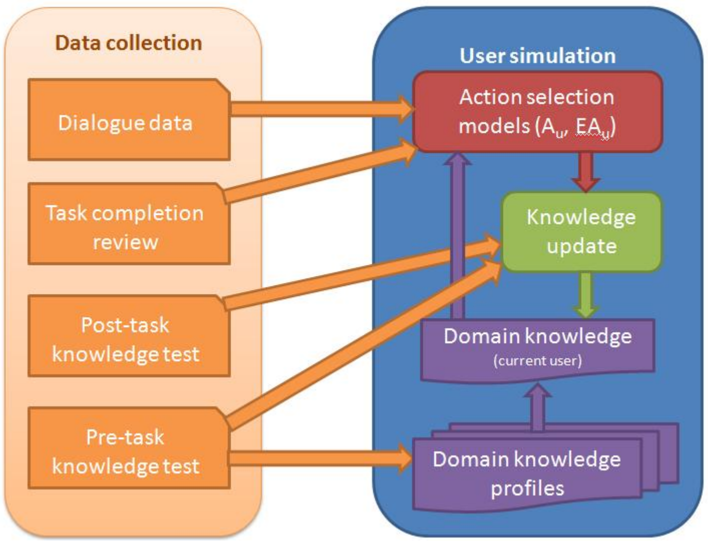


Figure 6.1: Populating our user simulation models with data

above: being sensitive to referring expressions. We do this by enabling the model to be able to ask for clarifications on referring expressions apart from other dialogue actions. Table 6.1 shows an example interaction between a user simulation model and the dialogue system at the dialogue action level. The dialogue system also passes its choice of referring expressions to the user simulation. One should note that actual literal expressions ($RE_i \in REC_{s,t}$) is not used in these transactions. We only use (R_i, T_i) which is a pair representing the referent (i.e. domain entity being referred to) and its type information (i.e. jargon or descriptive). Therefore $RE_i = (R_i, T_i)$. You can see such pairs in the given example interaction, where the referring expression choices of the system is represented in parenthesis following the dialogue action.

6.3.1.1 Dynamic Bayesian model

A simple approach to model real user behaviour that is sensitive to referring expressions (as in requirement 1) is to model user responses (dialogue act and environment act) on contextual information available in our corpus. These include all referring expressions used in the system's utterance, the user's current knowledge of the REs and the system's dialogue act. This will ensure that the responses are based on the user's domain knowledge and the referring expression types used by the system. Dynamic Bayesian models were earlier proposed by Pietquin and Dutoit (2006), in which the user's responses were based on their goals as well as the system actions. We modify that model to include system RE choices ($REC_{s,t}$) and user's domain knowledge of

```
SYS: connect(broadband_cable, livebox_adsl_socket);
       {(broadband_cable,jargon),(adsl_socket,jargon)}
USR: request_clarification((broadband_cable,jargon))
SYS: provide_clarification(broadband_cable);
       {}
USR: request_clarification((adsl_socket,jargon))
SYS: provide_clarification(adsl_socket)
USR: acknowledge_instruction
SYS: connect(broadband_cable, broadband_filter);
       {(broadband_cable,jargon),(broadband_filter,descriptive)}
```

Table 6.1: An example interaction between user simulation and dialogue system

jargon expressions ($DK_{u,t}$) as follows.

$$P(A_{u,t}|A_{s,t}, REC_{s,t}, DK_{u,t})$$

$$P(EA_{u,t}|A_{s,t}, REC_{s,t}, DK_{u,t})$$

However, with so much contextual information, there are data sparsity problems because many contexts are not seen in the small amount of data that we have in our corpus. For instance, all the dialogues in the corpus followed the *one-strategy-per-dialogue* rule. Therefore, there is more than one referent in the utterance, all of them are either jargon or descriptive. There is no mixture of jargon and descriptive expressions in any utterance in the corpus, which is precisely what we need in our experiment. In other words, although this model is *sensitive* to REs used by the system, it is not *trainable* using non-adaptive dialogues. Due to these problems, the dynamic Bayesian model cannot be used in its current format in our problem.

6.3.1.2 A Three-step pipeline model

We propose a modified version of the dynamic Bayesian model called the *three-step model*, in which the simulation of a user's response is divided into three steps:

1. Review all the referring expressions used by the system.
2. Interact with the simulated environment.

3. Respond to the system's instruction.

The user simulation (US) receives the system action ($A_{s,t}$) and its referring expression choices ($REC_{s,t}$) at each turn. For example,

$$\begin{aligned} A_{s,t} &= manipulate(connect(broadband\_cable, livebox\_asdl\_socket)) \\ REC_{s,t} &= \{(broadband\_cable, jargon), (livebox\_adsl\_socket, descriptive)\} \end{aligned}$$

System utterance = "Please connect one end of the broadband cable to the square socket with the gray stripe"

Please note that the system utterance (as in the above example) is not an input to the user simulation model. It is presented here only to illustrate how the information in $A_{s,t}$ and $REC_{s,t}$ translate into a system utterance. However, in real user evaluation, the user will be presented with a system utterance.

The US responds with a user action $A_{u,t}$. This can either be a clarification request (cr) or an instruction response (ir). In the first step, the simulation processes all the referring expressions used by the system ($REC_{s,t}$). We process the referring expressions individually and not for the whole set of expressions ($REC_{s,t}$) at once. Please note that the data at our disposal are non-adaptive dialogues between users with different levels of domain expertise and a dialogue system using either jargon or descriptive expressions. Therefore, when there is more than one referent in the utterance, the data collection dialogue system only produced the same type of expressions for all referents. However, to learn adaptive policies for referring expression generation, user simulation models have to be able to interact with an adaptive system (or a system that learns to be adaptive) that uses both types of expressions (for different referents) within the same utterance. Therefore, we propose a *RE recognition model* that processes the referring expressions individually for each referent and is thereby able to process any combination of RE types and respond appropriately to each referring expression. Therefore, this model can be trained using non-adaptive data that we have and still be used to interact with adaptive dialogue systems. In the subsequent steps, it deals with the environment action and responds to the dialogue system's instruction. The following are the three steps in this user simulation.

Step 1: The RE recognition model returns a clarification request based on the referring expression used and the user's domain knowledge ($DK_{u,t}$). We iterate this model for producing clarification requests for each referring expression used in the system's utterance ($\forall RE_i \in REC_{s,t}$). The model returns none when no clarification is produced. Therefore, when there are n referring expressions used by the system in an utterance, this model is used n times and clarification requests may be produced for any of the referring expressions in the list. If a clarification request is produced, it is presented to the system as the user's response. If no clarification is produced, then the *environmental action* model is used. The probability of the user's dialogue action ($A_{u,t}$) being a clarification request on an expression RE_i (i.e. $cr(R_i,T_i)$) is given by the following model.

$$P(A_{u,t} = cr(R_i,T_i)|R_i,T_i,DK_{u,t}(R_i))$$

For example, the user dialogue action *request_clarification((broadband_filter,jargon))* may be produced by the above statistical model, because the user did not know the jargon expression *broadband filter* in the system's choice of expressions. We use $DK_{u,t}(R_i)$ to represent the user's knowledge of the jargon expression for the referent R_i. Please note that clarifications are requested not based on the system's dialogue action but only on its referring expressions. The above model simulates the process of interpreting and resolving the each expression RE_i in $REC_{s,t}$ and identifying the domain entity of interest in the instruction. This model is, therefore both *sensitive* to the REs and by processing the REs individually, it is also *trainable*.

Step 2: In this step, we simulate the user's interaction with the environent. The environment contains domain entities which the user can either observe or manipulate. The *environment action* model simulates the user's environment actions that include either reporting the status of an observed entity to the system or manipulating them and acknowledging the instruction. There are several distinct environmental actions that a user can take following the system's instruction. However, we employ a simple model that classifies the user's physical action as either *correct* or *incorrect*. The model uses these classes to respond to the system's instruction. Its responses are therefore *correct* or *incorrect* action. When no clarification request is produced (i.e. $A_{u,t} == none$) after processing all the referring expressions, the environment action $EA_{u,t}$ is generated by the following model based on the system's dialogue action ($A_{s,t}$).

$$P(EA_{u,t}|A_{s,t})$$

Please note that no environmental action is produced when there is a clarification request. In such cases, $EA_{u,t} = none$.

Step 3: Finally, we simulate the user's response to the system's instruction. We use a *instruction response model* to produce the user's response ($A_{s,t}$). Here, the user's dialogue action can be either $provide\_info$, $acknowledgement$ or $other$ based on the system's instruction. We denote them summarily as *instruction response*. The probability of user's dialogue action being an instruction response (*ir*) is generated based on the system's action ($A_{s,t}$) and the user's environment action ($EA_{u,t}$) using the following model.

$$P(A_{u,t} = ir|EA_{u,t}, A_{s,t})$$

Please note that steps 2 and 3 are executed only when there are no request for clarifications generated in step 1. Figure 6.2 presents the flow chart of the three step model. The advantage of the three-step model over the dynamic Bayesian model is that it simulates real users in contexts that are not directly observed in the dialogue data. For example, the dialogue data does not contain a user's response to a mix of RE types. However, our model can respond appropriately in such contexts. The model will therefore respond to system utterances containing a mix of REG strategies (for e.g. one jargon and one descriptive expression in the same utterance). Such combinations of strategies are common in adaptive dialogues, where the system might decide to use jargon for one referent and descriptive expression for another depending on its knowledge of the user's expertise. In future, this model can be extended to an n-step model with one step for each kind of clarification request as classified by Schlangen (2004).

6.3.1.3 Class-based Three-step pipeline model

Although the three-step model is closer to real user behavior in terms of Kullback-Leibler divergence (see section 6.7), we observed some undesirable behaviour in the model that would make learning very difficult in our framework. This was due to the limitations in the data that we collected. There were many entities for which the probability of raising a clarification was low, even when the system used jargon and the user didn't know them. This is because of two shortcomings in our data collection set up:

- The system gave contextual clues for identifying some of the entities that the users didn't have to ask for clarifications. For example, when the users were asked to pick up a broadband filter, the system also said that there were two of

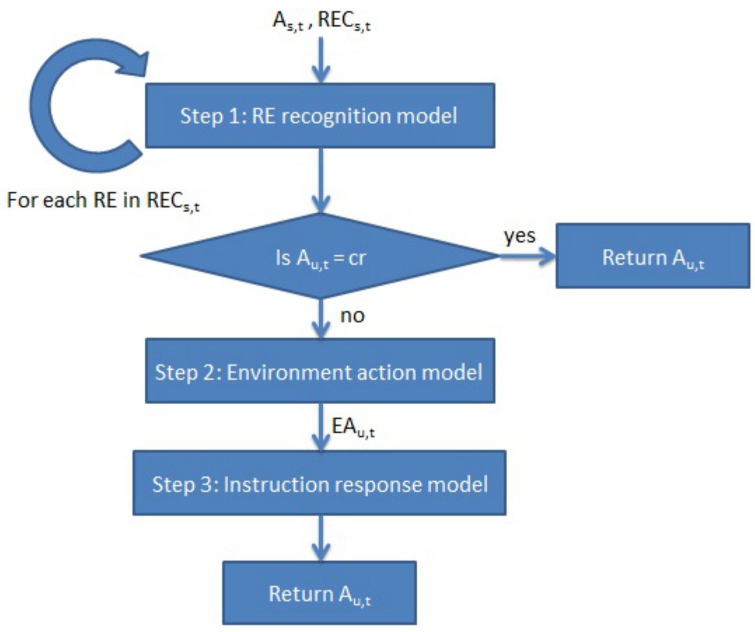

$A_{s,t}$, $REC_{s,t}$
Step 1: RE recognition model
For each RE in $REC_{s,t}$
Is $A_{u,t}$ = cr
yes
Return $A_{u,t}$
no
Step 2: Environment action model
$EA_{u,t}$
Step 3: Instruction response model
Return $A_{u,t}$

Figure 6.2: Flowchart of the three step user simulation model

them. Therefore even though users didn't know what broadband filters were, they recognised it because there were two of them in the box.

- Some of the entities were marked clearly using labels like "ADSL socket" (see figure 6.3)

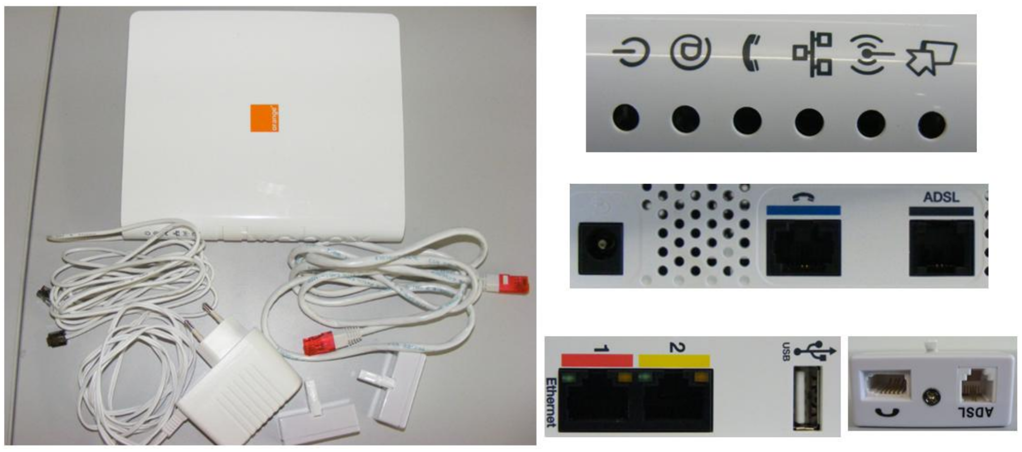


Figure 6.3: Issues in user environment during data collection

This problem affects policy learning because the fewer the clarification requests, the less the system gets to know about the user and therefore adapt to the user. So, in order to prevent these problems from affecting learning, we combined the domain entities into two classes: *hard* and *easy*.

- *Hard class*: This consists of domain entities that confuse the users when referred to using jargon expressions. The confusion could be because the users do not know the jargon expressions and it may be compounded when there are no other contextual clues for resolving the referent.
- *Easy class*: This consists of domain entities that users could easily recognise based on their knowledge or using contextual clues from the environment or from the system's utterances.

We classified them based on the number of clarification requests raised on the jargon references to these entities in our corpus. The mean number of clarification requests per referent was found to be 2.19. We classified those entities whose number of CRs is greater than or equal to the mean as members of the *hard* class and others as members of the *easy* class. The domain entities and their respective classes are given in table 6.2. Classifying the domain entities into two classes helps to solve the problem we faced in the three-step model. It allows us to move some entities between classes. For instance, a broadband filter which was recognised by most users can be classified as a member of the *hard* class based on the intuition that, although in the data collection setup, the broadband filter was easily recognised by all users due to contextual cues, in the final evaluation set up we could make it confusable and hard to recognise by not providing any cues. Therefore, we could move such domain entities that we deem to make hard to recognize in the evaluation process from the *easy* class to the *hard* class. We further describe this move in Chapter 7.

The model produces a clarification request *cr* based on the class of the referent ($C(R_i)$), type of the referring expression (T_i), and the current domain knowledge of the user ($DK_{u,t}(R_i)$). Clarification requests are produced as user's dialogue actions using the following *RE recognition model.*

$$P(A_{u,t} = cr(R_i, T_i)|C(R_i), T_i, DK_{u,t}(R_i))$$

The environment action and the instruction response are produced using the same models as in the three-step model when there is no clarification request produced in the first step. According to the data, clarification requests are much more likely when jargon expressions are used to refer to the referents that belong to the *hard* class and

Easy	**Hard**
Livebox	Ethernet cable
Wall phone socket	Broadband cable
Livebox ADSL socket	Livebox Broadband light
Livebox Power light	Livebox Ethernet light
Broadband filter	PC Ethernet socket
Livebox Ethernet socket	Power Adaptor
Livebox Power socket	

Table 6.2: Domain entities and their classes

which the user doesn't know about. When the system uses expressions that the user knows, the user generally responds to the instruction given by the system. We re-trained these models using the same dialogue data after annotating them with classes of referents.

Although the class-based model was developed to address the problems in our data, it is a useful model in its own right. It will always be possible to classify referents into several classes based on different features. In this work, we classified them into two groups based on the clarification requests in our data. However, other classifications are also possible. For instance, they can also be clustered into several classes based on how they are related to each other and a user's responses towards all the members could be similar. Therefore, we believe that wherever referents can be meaningfully classified and users' responses towards the members of a class are similar, the class-based model can be used. This is because the class-based model has a clear advantage of being able to handle data sparsity.

6.3.1.4 Training Action Selection models

We trained action selection models using our dialogue data using *relative frequency estimation*. We used the 12 dialogues (i.e. Jargon and Descriptive strategy) in our corpus. We avoided the tutorial strategy dialogues, because users were frustrated (see table 5.5) with the use of tutorial expressions which were longer than jargon and descriptive expressions. Moreover, jargon expressions produced learning gains as much as tutorial expressions (see table 5.5). We therefore planned to use only jargon and descriptive expressions as system choices and to train the action selection models only on jargon and descriptive strategy dialogues in the corpus. The RE recognition model of the three-step action selection model was trained from the following 4-tuple extracted from the dialogue corpus for each referring expression used by the system in every dialogue turn:

$$< R, T, DK_u(R), A_u >$$

$$\text{e.g.1. } < \mathit{broadband\_filter},\ \mathit{jargon},\ \mathit{no},\ \mathit{request\_clarification} >$$

$$\text{e.g.2. } < \mathit{livebox},\ \mathit{jargon},\ \mathit{no},\ \mathit{none} >$$

Where T is the type of expression used (jargon/descriptive), R is the domain entity (referent), $DK_u(R)$ is the user's knowledge of the jargon expression for R and A_u is the user's response (clarification request or none). User responses that are not clarification requests were first annotated as *none*. Therefore, *user response* in the above tuple was either a clarification requested on the referring expression or none. There were 203 jargon expressions and 202 descriptive expressions used by the system in our corpus which produced 405 tuples in the above format. We annotated the domain knowledge of the user based on their *pre-task recognition test* scores. For the class-based three step model, the referents were replaced by their classes. We use the following 4-tuple to train the RE recognition model of the class-based three step model:

$$< C(R), T, DK_u(R), A_u >$$

$$\text{e.g.1. } < \mathit{hard},\ \mathit{jargon},\ N,\ \mathit{request\_clarification} >$$

$$\text{e.g.2. } < \mathit{easy},\ \mathit{jargon},\ N,\ \mathit{none} >$$

Where $C(R)$ is the class of the domain entity (referent), T is the type of expression used (jargon/descriptive), $DK_u(R)$ is the user's knowledge of the jargon expression for R and A_u is the user's response (clarification request or none).

The environmental action and instruction response models were trained on a 3-tuple extracted from the corpus. These were extracted from dialogue turns where user responses were not clarification requests, as they have been accounted for in the RE recognition model. The information on user's manipulation of the environment comes from the task completion reports. Since we were not able to observe the user's observation actions on the domain entities, we annotated them manually depending on their response to the system. Both the manipulation and observation actions were manually annotated as either *correct* or *incorrect* based on how appropriate they were to the system instruction. Therefore, the three-tuple used to train the environmental action and instruction response models is:

$$< A_s, EA_u, A_u >$$

$$\text{e.g.} < manipulate(connect(broadband\_cable, broadband\_filter)),$$
$$correct,\ acknowledge >$$

Where A_s is the system's dialogue action, EA_u is the user's environmental action, A_u is the user's response.

6.3.2 Domain knowledge profiles

Complementary to the user action selection models presented in the previous section, we also model the domain knowledge of the users in the simulation module. This fulfills our second requirement of simulating a diverse population of users with different domain knowledge levels. During training, the dialogue system interacts with the user simulation several times producing several dialogues. At the start of each dialogue, we set the initial knowledge of the user simulation to one of the several knowledge profiles. Once instantiated, the simulation produces a dialogue behaviour that is consistent with its knowledge profile. For example, it behaves like a novice user asking a lot of clarification requests on jargon expressions when instantiated with a novice profile. The initial knowledge base ($DK_{u,initial}$) for 5 different users is shown in table 6.3. A novice user knows only "wall phone socket", and an expert knows all the jargon expressions. Between these two extremes, there are three intermediate profiles as well.

The knowledge of a jargon expression x is represented by the corresponding domain entity (R_x) in the profile. We use Y in the table to denote the user's knowledge of the jargon expression. The model assumes that users can interpret the descriptive expressions and resolve their references. Therefore, they are not explicitly represented. We use these five stereotype knowledge profiles as initial domain knowledge levels of users simulated during training and evaluation phases (see chapter 7). By using different profiles, the models produce a consistent behaviour of different types of users at various domain expertise levels.

Initially, the knowledge profiles from the pre-task knowledge test (see chapter 5) were clustered using the *k-means* clustering algorithm into 5 clusters to produce stereotypical knowledge profiles of users with different domain expertise levels (MacQueen (1967)). We used the centroid (i.e. center of the cluster) profiles as our stereotypes. However, as we pointed out in section 6.3.1.3, users (i.e. participants) were able to recognise some domain entities based on contextual cues provided by the system. We therefore manually modified the profiles from the clustering algorithm to suit a

	Novice	Int1	Int2	Int3	Expert
Phone socket	Y	Y	Y	Y	Y
Livebox		Y	Y	Y	Y
Livebox Power socket		Y	Y	Y	Y
Livebox Power light		Y	Y	Y	Y
Power adaptor			Y	Y	Y
Broadband cable				Y	Y
Ethernet cable			Y	Y	Y
Livebox Broadband light					Y
Livebox Ethernet light				Y	Y
Livebox ADSL socket					Y
Livebox Ethernet socket				Y	Y
PC Ethernet socket			Y	Y	Y
Broadband filter					Y

Table 6.3: Domain knowledge of 5 different users

more challenging environment (that we describe in Chapters 7 and 8) than the one used in data collection. For instance, novices and intermediates were able to identify an "ADSL socket" during data collection pre-test because the socket was labelled "ADSL". Similarly, they also recognised "broadband filter". However, in our final evaluation we remove such labels and therefore only experts will be able to identify an ADSL socket based on how it appears. Therefore, "ADSL socket" and "broadband filter" were marked unknown to all user types except experts. This revision of profiles required a small amount of domain expertise. The revised profiles are presented in table 6.3. Ideally, when the final evaluation environment is the same as the data collection environment, such revisions may not be necessary.

6.3.3 Knowledge update module

Corpus data shows that users can learn to associate new jargon expressions with domain entities during the conversation. We model this using the *knowledge update model*. This satisfies our third requirement of producing a learning effect and a dialogue behaviour that is consistent with an evolving domain knowledge of the user. For instance, a user who has been clarified on a jargon expression is more likely to have learned it and so would not thereafter ask for clarifications on the same expression later on in the conversation. Such behaviour is not possible without a learning model. We therefore model the user's domain knowledge DK_u to be dynamic which can be updated during the conversation. The domain knowledge is updated based on two types of system dialogue actions.

We observed in the dialogue corpus that users always learned a jargon expression when the system provides the user with a clarification. Therefore, the knowledge update is modelled using the following update rule.

$$\begin{gathered} if\ (A_{s,t} == provide\_clarification(R_x)), \\ then\ DK_{u,t+1}(R_x) \leftarrow 1 \end{gathered}$$

Users also learn when jargon expressions are repeatedly presented to them. Learning by repetition follows a non-linear learning curve - the greater the number of repetitions, the higher the likelihood of learning. This probabilistic update is modelled as a function of the referent (R_x) and a repetition parameter ($n(x)$) as follows (where $x \in REC_{s,t}$) .

$$P(DK_{u,t+1}(R_x) \leftarrow 1) = f(R_x, n(x))$$

The final state of the user's domain knowledge ($DK_{u,final}$) may therefore be different from the initial state ($DK_{u,initial}$) due to the learning effect produced by the system's use of jargon expressions. We trained the above model from our corpus based on how many times jargon expressions were repeated and users' post-task recognition scores.

6.4 Baseline models

We developed the following baseline action selection models to compare the performance of our pipeline models in simulating real users' dialogue behaviour.

6.4.1 Bigram model

A simple bigram model was built using the dialogue data by conditioning the user responses only on the system's dialogue act (Eckert et al. (1997)).

$$P(A_{u,t}|A_{s,t})$$

$$P(EA_{u,t}|A_{s,t})$$

Since it ignores all the context variables except the system dialogue act, it can be used in contexts that are not observed in the dialogue data. It also satisfies our first requirement that it must be sensitive to referring expressions by requesting for clarifications. However, this model is not conditioned on a knowledge profile. Its clarification request pattern may not be similar to the ones in our data collection. For instance, it may ask for a clarification for a jargon expression already clarified by the system.

6.4.2 Trigram model

The trigram model is similar to the bigram model, but with the previous system dialogue act $A_{s,t-1}$ as an additional context variable.

$$P(A_{u,t}|A_{s,t},A_{s,t-1})$$

$$P(EA_{u,t}|A_{s,t},A_{s,t-1})$$

With a little bit of history, this model may produce more sensible dialogue moves of the user. However, it has the same disadvantages of the bigram model like not being able to take the user's dynamic domain knowledge into account.

6.4.3 Equal Probability model

The equal probability model is similar to the bigram model, except that it is not trained on the dialogue data. Instead, it assigns equal probability to all possible responses for the given system dialogue act.

6.5 Smoothing

We used Witten-Bell discounting to smooth all the above models, in order to account for unobserved but possible events in dialogue contexts (Witten and Timothy (1991)). Witten-Bell discounting extracts a small percentage of probability mass, i.e. number of distinct events observed for the first time (T) in a context, out of the total number of instances (N), and redistributes this mass to unobserved events in the given context ($V-T$) (where V is the number of all possible events) . The discounted probabilities P^* of observed events ($C(e_i)>0$) and unobserved events ($C(e_i)=0$) are given below.

$$P^*(e_i) = \frac{C(e_i)}{N+T} \ if(C(e_i)>0)$$

$$P^*(e_i) = \frac{t}{(N+T)(V-T)} \quad if(C(e_i) = 0)$$

On analysis, for our use case we found that Witten-Bell discounting assigns greater probability to unobserved events than to observed events, in cases where the number of events per context is very low. For instance, in a particular context, the possible events, their frequencies and their original probabilities were - `provide _info` (3, 0.75), `other` (1, 0.25), `request _clarification` (0,0). After discounting, the revised probabilities P^* are 0.5, 0.167 and 0.33 respectively. `request _clarification` gets the whole share of extracted probability as it is the only unobserved event in the context and is more than the `other` event actually observed in the data. This is counter-intuitive for our application. Therefore, we use a modified version of Witten-Bell discounting (given below) to smooth our models, where the extracted probability is equally divided amongst all possible events. Using the modified version, the revised probabilities for the illustrated example are 0.61, 0.28 and 0.11 respectively.

$$P^*(e_i) = \frac{C(e_i)}{N+T} + \frac{T}{(N+T)V}$$

Other smoothing methods like *add-one* and *Good Turing* are only effective when there is a large corpus with a sizeable number of unique observed events (token count) per frequency (Church and Gale (1991)). From the above example, however, one can see that in the data at our disposal, this is not the case. There is one event that never occurs, one that occurs once and one event that has frequency three. Applying Good Turing smoothing to the above problem will re-estimate the frequencies of the events whose frequencies were originally zero and one as one and three respectively (because their relative frequency ratio is one). This is also counter-intuitive. Therefore, these methods were not used for smoothing.

6.6 Metrics for evaluation of simulations

Kullback-Leibler (KL) divergence, which is also called *relative entropy*, is a measure of how similar two probability distributions are (Kullback and Leibler (1951); Kullback (1959, 1987)). It can be used to measure how similar the distributions of the simulation models are to the real human user data. Several recent studies have used this metric to evaluate how closely their user simulation models produce real user behaviour (Cuayahuitl et al. (2005); Cuayahuitl (2009); Keizer et al. (2010)).

Model	$A_{u,t}$	$EA_{u,t}$
Dynamic Bayesian	1.25	0.97
Bigram	0.916	0.290
Trigram	0.981	0.301
Equal Probability	4.187	1.342
Three-step	0.862	0.274
Class-based three-step	0.711	0.232

Table 6.4: Dialogue Similarity (after smoothing)

We measure *dialogue similarity* (DS) based on Kullback-Leibler (D_{KL}) divergence between real and simulated dialogues. Since KL divergence is a non-symmetric measure, DS is computed by taking the average of the KL divergence between the simulated responses and the original responses (i.e. $D_{KL}(simulated||real)$) and vice versa (i.e. $D_{KL}(real||simulated)$). Dialogue Similarity (DS) between two models P and Q is defined as follows:

$$D_{KL}(P||Q) = \sum_{i=1}^{M} p_i * log(\frac{p_i}{q_i})$$

$$DS(P||Q) = \frac{1}{N} \sum_{i=1}^{N} \frac{D_{KL}(P||Q) + D_{KL}(Q||P)}{2}$$

The metric measures the divergence between distributions P and Q in N different contexts with M responses per context. Ideally, the dialogue similarity between two similar distributions is close to zero. Please note that in this evaluation, we do not divide the corpus into training and testing data.

6.7 Evaluation results

We compared the probability distributions of all the smoothed action selection models to the probability distribution of real user responses from our corpus using the `dialogue similarity` measure. The results of the evaluation are given in table 6.4.

Results show that both three-step models are closer to real user data than the baseline models. This is due to the fact that the bigram and trigram models don't take into account factors such as the user's knowledge, the strategy used, and the dialogue history. By effectively dividing the RE processing and the environment interaction, the three-step simulation model is not only realistic in observed contexts but also usable in unobserved contexts. We also show that the class-based three-step model is closer to real data when compared to all the other models. It is even better than the (RE-based) three-step simulation model. The class-based model is a generalised version of the three-step model and therefore must be more divergent from real user data than the three-step model. However, it is less divergent because the class-based three-step simulation essentially compresses the number of contexts because of classifying the domain objects into two classes and with fewer contexts, generalisation produced by smoothing techniques have less impact on the class-based model than the other models.

It should be noted that dialogue similarity is measured at the turn level by comparing the real and simulated responses in different contexts at each dialogue turn. However, simulated user responses have to be consistent across turns within a dialogue based on user's dynamic domain knowledge profile. Out of all the models presented in this chapter, only the three-step models produce a knowledge consistent behaviour because they take into account the user's domain knowledge at all times. Although the dynamic Bayesian model accounts for the user's knowledge it is unusable due to data sparsity issues in adaptive dialogue contexts.

6.8 Conclusion

We have presented data-driven user simulation models: the three-step and the class-based three-step models that satisfy our requirements for learning user modelling policies for adaptive REG using reinforcement learning. We have also shown that the three-step models are much closer to real user data than the other baseline models. By reviewing the referring expressions used in the system's utterance individually (and not as a set) and by the use of knowledge profiles and a learning model, these two models can be trained using limited non-adaptive dialogues and still be used to train and test adaptive systems. In chapter 7, we show how the class-based three-step model was used to train and test UM policies for adaptive referring expression generation.

Chapter 7

Learning Adaptive UM policies

In this chapter, we present a data-driven reinforcement learning framework to learn a user modelling policy for adaptive REG using the data-driven user simulations presented in chapter 6. We show that in comparison to hand-coded adaptive baseline policies the learned policies perform significantly better, with better adaptation accuracy over baseline policies. This is because the learned policy can adapt online to changing evidence about the user's domain expertise. We also compared the performance of policies learned using hand-coded and data-driven simulations and show that data-driven simulations produce better policies than hand-coded ones.

Section 7.1 describes the dialogue system framework and its modules. Section 7.2 describes the user simulation models. In section 7.4, we present the training results and in section 7.5, we present the testing results for different adaptive UM policies.

7.1 Self-Help Dialogue System

In this section, we describe the different modules of the dialogue system. The interaction between the different modules is shown in figure 7.1 (in learning mode). The dialogue system presents the user with instructions to set up a broadband connection at home. Please note that the dialogue task is not exactly the same as the one described in the basic framework (in chapter 4). In the Wizard of Oz setup that was used for data collection (in chapter 5) and for real user evaluation (in chapter 8), the system and the user interact using speech. However, in our machine learning setup which we present in this chapter, they interact with user simulation models at the abstract level of dialogue actions and referring expressions. Our objective is to learn to choose the appropriate referring expressions to refer to the domain entities in the instructions.

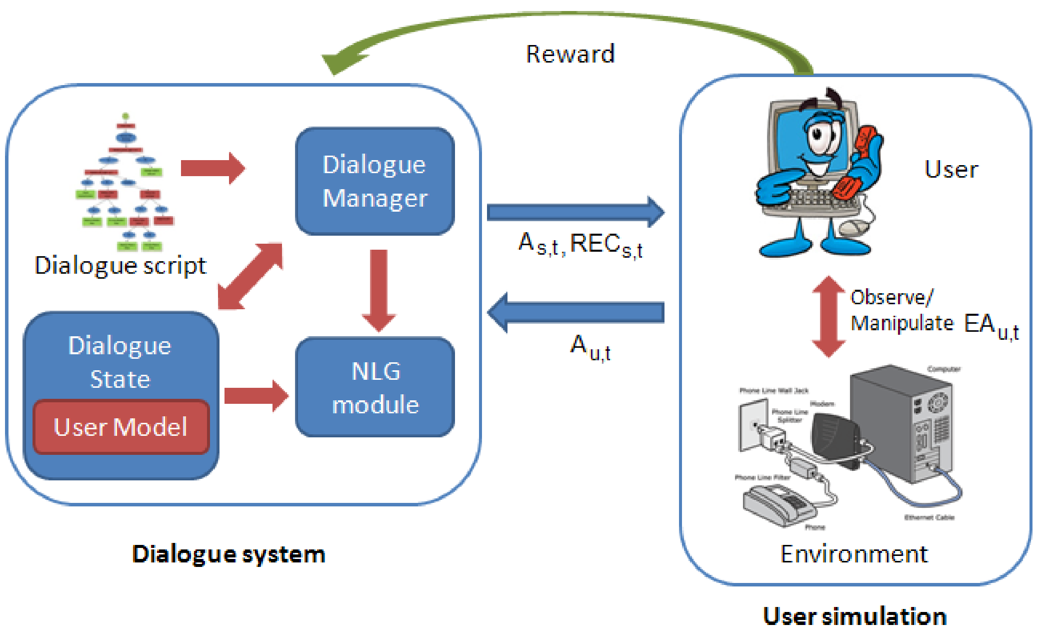


Figure 7.1: System User Interaction (in learning mode)

7.1.1 Dialogue Manager

The dialogue manager identifies the next instruction (dialogue act) to give to the user based on the dialogue management policy π_{dm}. Since, in this study, we focus only on learning the user modelling policy, the dialogue management is coded in the form of a finite state machine. In this dialogue task, the system provides two kinds of instructions - observation and manipulation. For observation instructions, users are supposed to observe a domain object in their environment and report back its status to the system, and for the manipulation instructions (such as plugging a cable in to a socket), they are supposed to manipulate the domain entities in the environment and acknowledge the instruction. When the user carries out an instruction, the system state is updated and the next instruction is given. Sometimes, users do not understand the referring expressions used by the system and then ask for clarification. In such cases, the system provides clarification on the referring expression ($provide\_clarification$), which is information to enable the user to associate the expression with the intended referent. The system action $A_{s,t}$ is therefore to either give the user the next instruction or a clarification. When the user responds in any other way, the instruction is simply repeated. The dialogue manager is also responsible for updating and managing the system state $S_{s,t}$(see section 7.1.2). The system interacts with the user by passing both the system action $A_{s,t}$ and the referring expressions $REC_{s,t}$ (see section 7.1.3).

7.1.2 Dialogue state

The dialogue state $S_{s,t}$ is a set of variables that represent the current state of the conversation. In our study, in addition to maintaining an overall dialogue state, the system maintains a user model $UM_{s,t}$ which records the initial domain knowledge of the user. It is a dynamic model that starts with a blank slate where the system does not have any idea about the user before the conversation starts. As the conversation progresses, the dialogue manager records the evidence presented to it by the user in terms of his dialogue behaviour, such as asking for clarification and interpreting jargon. Since the model is updated according to the user's behaviour, it may be inaccurate if the user's behaviour is itself uncertain. So, when the user's behaviour changes (for instance, from novice to expert), this is reflected in the user model during the conversation. Hence, unlike previous studies mentioned in chapter 2, the user model used in this system is not always an accurate model of the user's knowledge and reflects a level of uncertainty about the user.

Each jargon referring expression `x` has two corresponding variables in the dialogue state: `user_knows_x` and `user_doesnt_know_x`. They are both initially set to 0. The variables are updated using a simple user model update algorithm. If the user responds to an instruction containing the referring expression x with a clarification request, then `user_doesnt_know_x` is set to 1. Similarly, if the user responds with appropriate information to the system's instruction, the dialogue manager sets `user_knows_x` to 1 and `user_doesnt_know_x` to 0. Therefore only 3 states are possible for each jargon expression (not 4, as one would expect with 2 binary variables). Since there are 13 entities and we only model the jargon expressions with 2 binary variables, there are 26 variables. Each pair of these variables takes only 3 out of 4 values, therfore the state space size is 3^{13} (approximately 1.5 million states).

The user's knowledge is inferred and update by the dialogue manager during the course of the dialogue from their behaviour after each turn. For instance, if the user asks for clarification on a referring expression, his knowledge of the expression is set to n and when no clarifications are requested, it is set to y. This rule has been used in the past to implicitly acquire information about a user's knowledge (Chin (1989)). The user may have the capacity to learn jargon. However, only the user's initial knowledge is estimated and represented in the internal user model. This is based on the assumption that an estimate of the user's initial domain knowledge helps to predict the user's knowledge of the rest of the referring expressions.

7.1.3 REG module

The REG module is a part of the NLG module whose task is to identify the list of domain entities to be referred to in the dialogue act and to choose the appropriate referring expression for each of the domain entities for each given dialogue act. In this study, we focus only on the production of appropriate referring expressions to refer to domain entities mentioned in the dialogue act. It chooses between the two types of referring expressions - jargon and descriptive. For example, the domain entity *broadband_filter* can be referred to using the jargon expression "broadband filter" or using the descriptive expression "small white box"[1]. We call this act of choosing the *REG action*. The tutorial strategy was not investigated here since the corpus analysis showed tutorial utterances to be very time consuming.

The REG module operates in two modes - learning and evaluation. In learning mode, the REG module is the learning agent. The REG module learns to associate dialogue states with optimal REG actions. This is represented by a UM policy $\pi_{UM} : UM_{s,t} \rightarrow REC_{s,t}$, which maps the states of the dialogue (user model) to optimal REG actions. The referring expression choices ($REC_{s,t}$) is a set of pairs identifying the referent R_x and the type of expression T_x used (where x is literally the referring expression). For instance, the pair $(broadband\_filter, desc)$ represents the descriptive expression "small white box". Therefore,

$$REC_{s,t} = \{(R_{x_1}, T_{x_1}), ..., (R_{x_n}, T_{x_n})\}$$

7.2 User Simulations

Two kinds of user simulation models were used for training: data-driven and hand-coded.

7.2.1 Class-based Three-step model

We use the class-based three-step model presented in section 6.3.1.3. The simulation has a number of in-built knowledge profiles that determine the dialogue behaviour of the user being simulated. The user's knowledge of a referring expression x or the ability of interpreting it affects its request for clarification. If the user is able to interpret all the

[1]We will use italicised forms to represent the domain entities (e.g. *broadband_filter*) and double quotes to represent the referring expressions (e.g. "broadband filter").

Easy	**Hard**
Livebox	Ethernet cable
Wall phone socket	Broadband cable
Livebox Power light	Livebox Broadband light
Livebox Power socket	Livebox Ethernet light
	PC Ethernet socket
	Power Adaptor
	Livebox ADSL socket
	Livebox Ethernet socket
	Broadband filter

Table 7.1: Domain entities and their classes : revised

referring expressions used by the system ($REC_{s,t}$) and identify the references then the model processes the system's instruction, interacts with the environment and responds to the system. These models were populated by data from our dialogue corpus.

The model classifies the referents in the domain into *hard* and *easy* classes. As mentioned in chapter 6, the use of classes provides the flexibility to move some domain objects from one class to another. We moved 3 domain entities from the *easy* class to the *hard* class under the assumption that during real user evaluation these entities will be made harder to recognise by avoiding contextual cues. We now have 9 domain entities that are harder to recognize and 4 entities that are easier to recognise. The revised list of domain entities and their classes are given in table 7.1. In chapter 8, we show how we revise our evaluation environment accordingly.

Three out of the five domain knowledge profiles were used for training: Expert, Novice and Int2 (an intermediate), while all five profiles were used during evaluation. The user domain knowledge is initially set to one of several profiles at the start of every conversation. There is also a *knowledge update* module that updates the domain knowledge of the simulated user when they learn new jargon expressions (see section 6.3.2).

7.2.2 Hand-coded user simulation

We also built another simulation using the above models but where some of the parameters were set by hand instead of estimated from the data. The purpose of this simulation is to investigate how learning with a data-driven simulation compares to learning with a hand-coded simulation. Here we modified the dialogue behaviour of the users by setting them manually. The knowledge patterns and learning models were the same as in the data-driven simulation. In the hand-coded simulation, however, the user *always* asks for a clarification when he does not know a jargon expression (regardless of the class of the referent) and never does this when he does knows it. This enforces a stricter, more consistent behaviour for the different knowledge patterns, which we hypothesize should be easier to learn to adapt to, but may lead to less robust UM policies.

7.3 Reward function

As discussed in the previous chapters, a reward function generates a numeric reward for the learning agent's actions. It gives high rewards to the agent when the actions are favourable and low rewards when they are not. In short, the reward function is a representation of the goal of the agent. It translates the agent's actions into a scalar value that can be maximized by choosing good action sequences.

7.3.1 User score

Several other parameters like learning gain, dialogue time, etc could be used as a part of the reward function based on the needs of the system designer. One of the systematic ways of designing reward functions is to use the PARADISE framework (Walker et al. (2000)). The PARADISE framework is generally used to construct a linear function to predict user satisfaction scores from objective parameters like dialogue time, learning gain, etc. Such a function could be used as reward function so that the agent learns to maximise users' satisfaction scores. The user score was calculated as the mean of the following 5 dimensions on which we asked the user to rate the system during data collection (see Chapter 5).

1. Easy to identify referents
2. Learned useful terms
3. Complexity of instructions
4. Conversation was of right length

5. Will use the system in future

Each of these were rated on a scale of 1 to 4 (1 - strongly disagree, 2 - disagree, 3 - agree, 4 - strongly agree). We used linear regression to derive a linear function for user score (US) based on (normalised) learning gain (LG), dialogue time (DT) and task completion rate (TCR) as given below ($R^2 = 0.49, P = 0.027$). We could therefore calculate the user score using the following linear function.

$$User\ Score\ (US) = 0.67 + 0.73 * LG + 0.28 * TCR - 0.08 * DT$$

However, we chose not to use user satisfaction scores as reward although we collected them as a part of our data collection. The primary reason for this is that the user satisfaction scores were not discriminative enough to identify adaptive dialogues from non-adaptive ones because the user score (US) does not have a component based on adaptivity of the system. This is because, during data collection, the strategies were never mixed within a single conversation. Therefore, the participants were never presented referring expressions in an adaptive fashion. Either they were presented one of the three types of expressions throughout the entire conversation as we did not have an adaptive strategy to start with. We also did not pick expressions randomly. This was done to avoid the user's getting frustrated by random choices of expressions. Due to this fact, no user score regarding adaptation was collected. Instead, users scored the systems based on other parameters. Novice users scored the system well because of learning gain and expert users scored it well due to low dialogue time. Due to this reason and due to limited data that we had at our disposal, we decided to instead use an objective parameter like accuracy of adaptation for reward function. We collected the user's feedback on the adaptive nature of the system during final evaluation with real users (see chapter 8), which also show that users do not score highly adaptive systems higher than less adaptive ones.

7.3.2 Accuracy of adaptation

Since we could not use user score as our reward function, we implemented another way to measure adaptation. We designed a reward function for the goal of adapting to each user's domain knowledge. We present the Adaptation Accuracy score (AA) that calculates how accurately the agent chose the expressions, with respect to the user's initial knowledge. Appropriateness of an expression is based on the user's knowledge of the expression. So, when the user knows the jargon expression for a referent (r), the appropriate expression to use is jargon, and if s/he doesn't know the jargon expression, a descriptive expression is considered appropriate. Although the user's domain knowledge is dynamically changing due to learning, we base appropriateness on the initial state, because our objective is to adapt to the initial state of the user ($DK_{u,initial}$). We calculate accuracy per referent (Acc_r) as the ratio of number of appropriate expressions to the total number of instances of the referent in the dialogue. We then calculate the overall mean accuracy (AA) over all referents as shown below.

$$Acc_r = \frac{\#(appropriate\_expressions(r))}{\#(instances(r))}$$
$$Adaptation\ Accuracy\ (AA) = \frac{1}{\#(r)} \Sigma_r Acc_r$$

Note that this reward is computed at the end of the dialogue (it is a 'final' reward), and is then back-propagated along the action sequence that led to that final state. Since the agent starts the conversation with no knowledge about the user, it may try to use more sensing moves to seek information about the user's domain knowledge. However, by measuring accuracy to the initial user state, the agent is encouraged to restrict its exploratory moves and start predicting the user's domain knowledge as soon as possible. The system should therefore ideally balance sensing and adaptation to increase accuracy. The above reward function returns 1 when the agent is completely accurate in adapting to the user's domain knowledge and it returns 0 if the agent's REC choices were completely inappropriate. Usually during learning, the reward value lies between these two extremes and the agent tries to maximize it to 1.

7.4 Training

The REG module was trained (operated in learning mode) using the above simulations to learn UM policies that select referring expressions based on the user expertise in the domain. As shown in figure 7.1, the learning agent (REG module) is given a reward at the end of every dialogue. During the training session, the learning agent explores different ways to maximize the reward. The REG module was trained in learning mode using adaptive accuracy (AA) as reward function. Using the SARSA reinforcement learning algorithm (with linear function approximation) (Sutton and Barto (1998); Rummery and Niranjan (1994); Shapiro and Langley (2002)), the training produced approx. 5000 dialogues. The exploration parameter was set to 0.3. This means that when the training starts, 30% of the time, the agent will choose to explore rather than exploit the Q-values. This value however decays gradually over time.

Two types of simulations were used as described above: Data-driven and Hand-coded. Both user simulations were calibrated to produce three types of users: novice, intermediate and experts. Novice users knew just one jargon expression, intermediates knew seven, and experts knew all thirteen jargon expressions. There were underlying patterns in their knowledge which were learned from the corpus data. For example, Intermediate users were those who knew the commonplace domain entities but not those specific to broadband connection. For instance, they knew "ethernet cable" and "pc ethernet socket" but not "broadband filter" and "broadband cable". The three profiles were picked up randomly. However they were picked with equal probability so that the agent would learn to adapt to different types of users without any bias towards users of any particular type.

Figure 7.2 shows how the agent learns using the data-driven (**Learned DS**) and hand-coded simulations (**Learned HS**) during training. It can be seen in figure 7.2 that towards the end the curve plateaus signifying that a policy has converged.

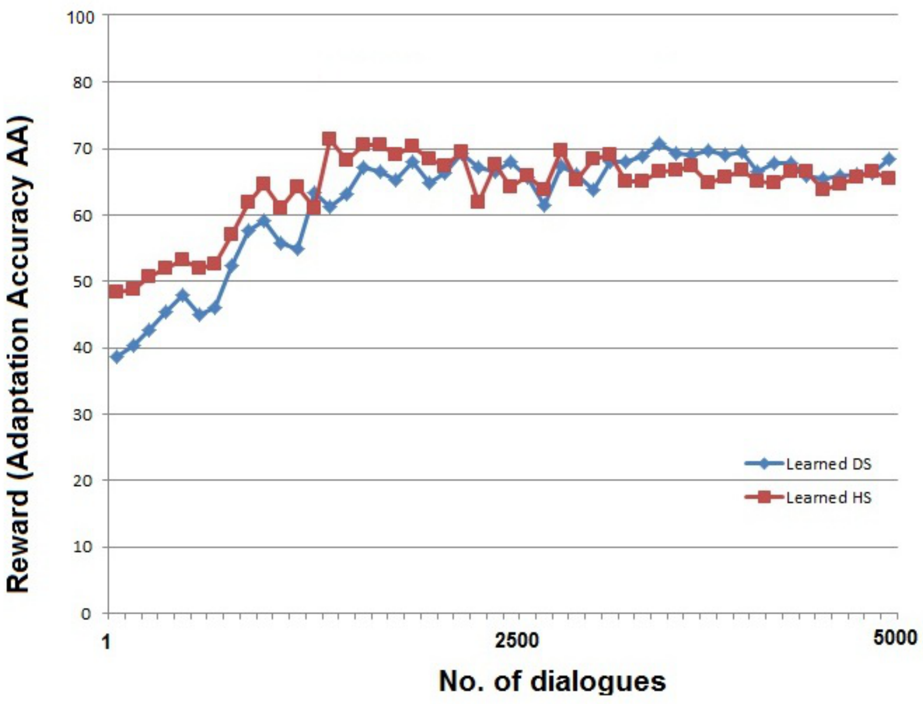


Figure 7.2: Policy Learning curves

Initially, the learning agent chooses randomly between the referring expression types for each domain entity in the system utterance, irrespective of the user model state. Once the referring expressions are chosen, the system presents the user simulation with both the dialogue act and referring expression choices. The choice of referring expression affects the user's dialogue behaviour which in turn makes the dialogue manager update the user model. For instance, choosing a jargon expression could evoke a clarification request from the user, which in turn prompts the dialogue manager to update the user model with the new information that the user is ignorant of the particular expression. The same process is repeated for every dialogue instruction. At the end of the dialogue, the system is rewarded based on its choices of referring expression. The reward is set to be proportional to how accurately (with respect to the user's true initial domain knowledge) the system chooses the referring expressions. So, if the system chooses jargon expressions for novice users or descriptive expressions for expert users, penalties are incurred and if the system chooses REs appropriately, the reward is high. On one hand, those actions that fetch more reward are reinforced and on the other hand, the agent tries out new state-action combinations to explore the possibility of greater rewards. The pace of learning is governed by the *learning rate* parameter and exploration is governed by a *exploratory parameter* which is halved periodically. Therefore as the exploratory parameter reduces to zero, the learning agent stops exploring new state-action combinations and exploits those actions that contribute to higher reward. The REG module learns to choose the appropriate referring expressions based on the user model in order to maximize the overall expected long-term reward.

7.5 Evaluation

We evaluate the learned policies with simulated users. Although it is ideal to test the learned policies with real users, evaluation with real users is an expensive process. Therefore it is a common strategy to first test the learned policies with simulated users and then evaluate them with real users (Lemon et al. (2006); Filisko and Seneff (2006); Frampton (2008); Rieser (2008)). In this section, we first present the evaluation metrics used, the baseline policies that were hand-coded for comparison, and finally, the results of evaluation. We test the policies with real users in chapter 8.

7.5.1 User simulation

In the evaluation mode, candidate policies interact with many unknown simulated users in the same way as in the learning mode. However we set the learning rate and exploration parameters to zero. The system consults the learned policy to choose the referring expressions based on the current user model. The data-driven simulation was calibrated to simulate 5 different user types. In addition to the three users - Novice, Expert and Int2, from the training simulations, two other intermediate users (Int1 and Int3) were added to examine how well the learned policies handle unseen user types (see Table 6.3 for all the knowledge profiles).

7.5.2 Metrics

In addition to the Adaptation Accuracy (AA) score mentioned in the reward function, we also measure other parameters from the conversation in order to show how learned adaptive policies compare with other possible policies on other dimensions. We calculate the time taken ($Time$) for the simulated user to complete the dialogue task. This is calculated using a regression model ($R^2 = 0.98, P = 0.000$) shown below from the corpus based on number of words ($\#(W)$), turns (T), and mean user response time (URT).

$$Dialogue\ Time\ (DT)(mins) = (19.75 + 0.6 * \#(W) + 0.78 * URT * T)/60.0$$

We also measure the (normalised) learning gain (LG) produced by using unknown jargon expressions. This is calculated using the pre and post scores from the user domain knowledge (DK_u) as follows.

$$Learning\ Gain\ LG = \frac{Post\% - Pre\%}{100 - Pre\%}$$

The time and learning gain parameters were used to later calculate simulated user scores (US) (see section 7.3.1).

7.5.3 Baseline UM policies

In order to compare the performance of the learned policy with hand-coded UM policies, four rule-based adaptive policies were built.

Descriptive: Uses descriptive expressions for all referents by default.

Jargon-adapt: Uses jargon for initial reference for all referents by default. But it changes to using descriptive expressions for those referents for which users asked for clarifications. See table 7.2 for an example dialogue.

Sys: Do you have a broadband cable in the package?
Usr: What is a broadband cable?
Sys: The broadband cable is the thin black cable with colourless plastic ends.
Usr: Yes. I have that.
....
Sys: Please plug one end of the thin black cable with colourless plastic ends into the broadband filter.

Table 7.2: Jargon-adapt policy: An example dialogue

Switching-adapt: This policy starts with jargon expressions for initial references and continues using them until the user requests for clarification. After a clarification request, it switches to descriptive expressions for all new referents and continues to use them until the end. In an example dialogue shown in table 7.3, please note that after the first clarification request it uses descriptive for the next referent onwards.

Stereotypes: In this policy, we use the knowledge profiles from our data collection. The system starts using jargon expressions for the first *n* turns and then based on the user's responses, it classifies them into one of the five stereotypes and thereafter uses their respective knowledge profiles in order to choose the most appropriate referring expressions. For instance, if after *n* turns, the user is classified as a novice, the system uses the novice profile to choose expressions for the referents in the rest of the dialogue. We tested various values for *n* with simulated users (see section 7.5.1) and used the one that produced the highest accuracy (i.e. n = 6). Please note that as the value of *n* goes up from 1, accuracy increases as it provides more evidence for classification. However, after a certain point the adaptation accuracy starts to stabilize, because too much sensing is not more informative. Later it starts to fall slightly because sensing moves come at the cost of adaptation moves. See table 7.4.

Please note that the Jargon-adapt and Switching-adapt policies exploit the user model in their subsequent references. When the system knows that the user does (or doesn't) know a particular expression, this knowledge is exploited in subsequent turns by using the appropriate expressions and therefore the system is adaptive. We distinguish this kind of adaptation from the kind learned by the system using the reinforce-

Sys: Do you have a broadband cable in the package?
Usr: What is a broadband cable?
Sys: The broadband cable is the thin black cable with colourless plastic ends.
Usr: Yes. I have that.
Sys: Do you have a small white box that has two sockets and a phone plug in the package?
....
Sys: Please plug one end of the thin black cable with colourless plastic ends into the small white box that has two sockets and a phone plug.

Table 7.3: Switching-adapt policy: An example dialogue

No of steps	Adatation Accuracy (%)
3	51.23
4	58.18
5	58.56
6	72.46
7	71.5
8	71.0
9	70.7
10	69.23
11	68.22
12	67.04

Table 7.4: Stereotypes: n-values and Adaptation Accuracy

Policies	AA (%)	DT (mins)	LG
Descriptive	46.15 (± 33.29)	**7.44**	0
Jargon-adapt	74.54 (± 17.9)	9.15	**0.97**
Switching-adapt	62.47 (± 17.58)	7.48	0.30
Stereotype (n=6)	72.46 (± 20.77)	8.15	0.49
Learned HS	69.67 (± 14.18)	7.52	0.33
Learned DS	**79.99** (± 10.46)	8.08	0.63

Table 7.5: Evaluation on 5 simulated user types

ment learning framework. We call the former *local adaptation* and the latter *global adaptation*.

7.5.4 Results

The REG module was operated in evaluation mode to produce around 200 dialogues per policy distributed equally over the 5 user groups. Overall performance of the different policies in terms of $AdaptationAccuracy(AA)$, $DialogueTime(DT)$, $LearningGain(LG)$ and $Userscore(US)$ are given in Table 7.5.

We found that the Learned DS policy (i.e. learned with the data-driven user simulation) is the most accurate (Mean = 79.99, SD = 10.46) in terms of adaptation to each user's initial state of domain knowledge. It outperforms all other policies: Learned HS (Mean = 69.67, SD = 14.18), Switching-adapt (Mean = 62.47, SD = 14.18), Jargon-adapt (Mean = 74.54, SD = 17.9), Stereotype (Mean = 72.46, SD = 20.77) and Descriptive (Mean = 46.15, SD = 33.29). The accuracy (AA) of the Learned DS policy and all other policies were compared using two-tailed paired t-test. Accuracy of adaptation of the Learned DS policy was significantly better than Descriptive policy (P = 0.000, t = 9.11, SE = 37.413), Jargon-adapt policy (P = 0.01, t = 2.58, SE = 20.19), Stereotype policy (P = 0.000, t = 3.95, SE = 23.40), Switching-adapt policy (P = 0.000, t = 8.09, SE = 22.29) and Learned HS policy (P = 0.000, t = 5.59, SE = 20.20). The Learned DS policy performs better than the Jargon-adapt policy, because it is able to predict accurately the user's knowledge of referents unseen in the dialogue so far and therefore adapts both locally and globally. It performs better than the Stereotype policy and Learned-HS policies as its adaptive behaviour takes into account the uncertainty in user's dialogue behaviour. The Learned-DS policy also continuously adapts to new evidence which contributes to its high performance.

The Jargon-adapt policy performs better than the Learned HS, Switching-adapt and Descriptive policies ($P < 0.05$). This is because the system can learn more about the user by using more jargon expressions and then use that knowledge for local adaptation. However, it is not possible for this policy to predict the user's knowledge of unseen referents and therefore does not adapt globally. Jargon-adapt performs slightly better than the Stereotype policy but increase in accuracy is not statistically significant ($P = 0.17$).

The Stereotype policy performs significantly better than the Switching-adapt and the Descriptive policies ($P < 0.001$) but not is not significantly different from the Learned-HS and Jargon-adapt policies. The Stereotype policy adapts to users globally using their profiles. However, due to uncertainty in user's responses, it is not always possible to pick the right profile for adaptation. This is probably why it beats the Switching-adapt and the Descriptive policies and performs as well as the Learned-HS and the Jargon-adapt policies.

Although the Learned HS policy is similar to the Learned DS policy, as shown in the learning curves in figure 7.2, it does not perform as well when confronted with user types that it did not encounter during training. Another reason for its poor performance is that it does not take into account the uncertainty of the user's dialogue behaviour. The Switching-adapt policy, on the other hand, quickly switches its strategy (sometimes erroneously) based on the user's clarification requests but does not adapt appropriately to evidence presented later during the conversation. Sometimes, this policy switches erroneously because of the uncertain user behaviours. The Descriptive policy performs very well with novice users but not so with other user types.

In terms of dialogue time (DT), learned policies are a bit more time-consuming than the Switching-adapt and Descriptive policies but less than the Jargon-adapt policy. This is because learned policies use sensing moves (giving rise to clarification requests) in order to learn more about the user. The Descriptive policy is non-adaptive and therefore scores better than other policies because it only uses descriptive expressions and therefore causes no clarification requests from the users. Similarly due to fewer clarification requests, the Switching-adapt policy also takes less dialogue time. Learned policies spend more time in order to learn about the users they interact with before they adapt to them. We argue that, although in this task where adaptation is only at the level of referring expressions, Descriptive and Switching-adapt policy may take less dialogue time than learned policies, in tasks where adaptation also happens at the level of dialogue management, adaptative systems will provide instructions of appropriate complexity and therefore save dialogue time more than its non-adaptive competitors. When the three high performing policies (by adaptation accuracy) are compared, the Learned-DS policy has the shortest dialogue duration. This is due to better adaptation. The difference between Learned-DS and Jargon-adapt policy is statistically significant ($P<0.05$). However the difference between Learned-DS and Stereotype policy is not significant.

With respect to (normalised) learning gain (*LG*), the Jargon-adapt policy produces the highest gain ($LG = 0.97$). This is because the policy used jargon expressions for all referents at least once. The difference between Jargon-adapt policy and others was statistically significant at $P < 0.0001$. The Learned DS produced a learning gain of 0.63 which is a close second because it did use jargon expressions with novice users until it was ready to adapt to them. While the use of jargon expressions with novices and intermediates sacrificed adaptation accuracy, it served to increase (normalised) learning gain apart from populating the user model.

Figure 7.3 shows how each policy performs in terms of accuracy on the 5 types of users. When compared in terms of how each policy performs with different user types, we found that the Descriptive policy performs very well with novice users. However, its performance drops with increasing user expertise. The Switching-adapt policy performs well with users on both extremes of the knowledge spectrum: experts and novices. This is because switching happens early in the dialogue for these user types. But for the three intermediate user types, its performance falls below 60%. The Learned-HS policy performs well with Novices, Intermediates 1 and 2, however as the number of clarifications reduce (i.e. Intermediate 3 and Experts), it is not able to adapt well them in the face of uncertainty. In contrast to these policies, the Stereotype policy performs well with all user types (around 60%-70% accuracy). This is because of global adaptation using user knowledge profiles. However, it performs worse than the Learned-DS policy because it is unable to handle uncertainty at the user's end. It therefore misclassifies users at times. The Jargon-adapt policy adapts well with Experts but its performance goes down as expertise decreases. However its performance is not as bad as the Descriptive policy as it adapts locally even with Novice users. The only policy that adapts very well with all user types is the Learned-DS policy. Its adapts very well with the user types its trained with (i.e. Novice, Intermediate 2 and Experts). It also adapts considerably well with those users that were unseen in during the training phase (i.e. Intermediates 1 and 3).

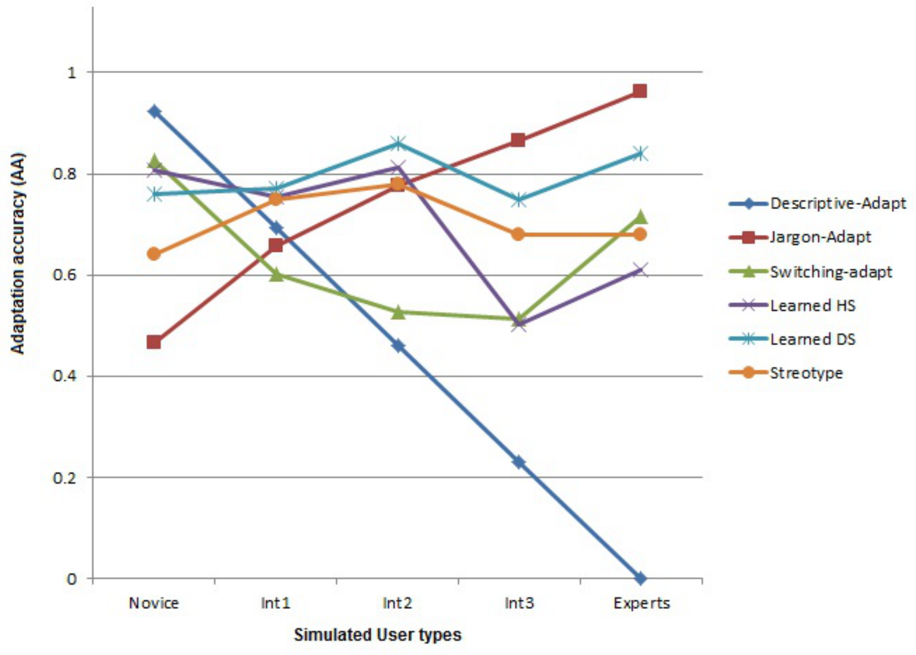


Figure 7.3: Evaluation - Adaptation Accuracy vs User types

7.5.5 Discussion

In this study, we have shown that the Learned DS policy:

1. Learns to adapt: The learned policies sense the user's expertise and predict their knowledge patterns, in order to better choose expressions for referents unseen in the dialogue so far. The system learns to identify the patterns of knowledge in the users with little sensing. So, when it is provided with a piece of evidence (e.g. the user knows "broadband filter"), it is able to accurately estimate unknown facts (e.g. the user might know "broadband cable"). Sometimes, its choices are wrong due to incorrect estimation of the user's expertise (due to the stochastic behaviour of the users). In such cases, the incorrect adaptation move is considered to be an sensing move to populate the user model. This helps further adaptation using the new evidence.

2. Learns to sense and populate the user model: In addition to adaptation, learned policies learn to identify when to sense information from the user to populate the user model (which is initially set to `unknown`). It should be noted that the system cannot adapt unless it has some information about the user and therefore needs to decisively sense information by using jargon expressions. If it senses information all the time, it is partly sacrificing adaptation to the user. The learned policies therefore learn to trade-off between information sensing moves and adaptive moves in order to maximize the overall adaptation accuracy score. By continuously using this sense-predict-adapt approach, the system adapts dynamically to different users. Therefore, with little sensing and better prediction, the learned policies are able to better adapt to users with different domain expertise.

3. Adapts better than hand-coded policies: Results show that Learned DS policy adapts better with all user types than all the hand-coded policies (See figure 7.3). Hand-coded policies were manually coded with no expertise in the domain at all. Therefore, some of the policies are biased towards one type of users or the other. Descriptive policy is biased to novice users. One can see its performance steadily declining as the expertise of the user types increases. Similarly, Jargon-adapt policy is biased more towards expert users. One can also see that it adapts to all types of users although locally. Switching-adapt policy is biased to both novice and expert users but performs badly with intermediates. This is because it is susceptible to erroneously switching strategies with intermediate users. The Stereotype policy performs well with all users but is susceptible to uncertain user dialogue moves. In contrast to all these policies, the Learned DS policy uses the information it has in the user model to predict the user's domain knowledge and choose appropriate referring expressions.

4. Adapts better than policy learned with hand-coded simulation: Results also show that the Learned DS policy adapts better to all users than the policy learned using the hand-coded user simulation (See table 7.5). Although the learning curves for both these policies look similar to each other (See fig. 7.3), they perform differently with different users during evaluation.

5. Genaralises to unseen users: This study has also shown that the Learned DS policy which was trained only using three out of five user knowledge profiles (Novice, Intermediate 2 and Expert), generalises to all five user types (See fig. 7.3). It adapts to other intermediate user types (int1 and int3) very well although not at the same level as the three profiles it was trained to adapt. In fact, it is the only policy that performs more or less uniformly with all users.

7.6 Conclusion

In this chapter, we have presented a data-driven framework to learn adaptive UM policies. We also showed that an adaptive user modelling policy can be learned using our new class-based three-step user simulation model and that such policies generalise to more users during testing than the ones used for training. Learned policies sensed the users' domain knowledge, modelled their knowledge patterns, and later adapted to them successfully by choosing the most appropriate referring expressions. We have shown that the policy learned using a data-driven simulation can significantly outperform both hand-coded adaptive policies and a policy learned using a hand-coded simulation. This framework can be employed where adapting to an unknown user's domain knowledge is very important, for example in automated technical support. In the following chapter, we show how learned policies perform for real users in comparison with hand-coded adaptive policies.

Chapter 8

Evaluation with Real Users

Do the learned policies perform as well with real users as they perform well with simulated users? Henderson et al. (2005, 2008) and Rieser (2008) showed that results from simulated user evaluation transfer successfully to real user evaluation. In this chapter, we evaluate the performance of two policies with real users in real environments. Users were given the task of setting up a broadband Internet connection while listening to instructions from our dialogue system. In a "wizarded" setup, similar to the one used in our data collection, users interacted with dialogue systems using one of the two user modelling policies. Their interactions were logged and analysed. We present the results of the evaluation in this chapter.

8.1 Candidates for evaluation

Real user evaluation is expensive. We therefore compared the performance of only two out of the six policies we evaluated with simulated users. We chose the top two performing adaptive policies from the simulated user evaluation: Jargon-adapt and Learned DS.

8.1.1 Jargon-adapt policy

The Jargon-adapt policy is a hand-coded policy whose behaviour is deterministic. It always chooses a jargon expression for every domain object when it is first mentioned. It continues to use jargon expressions for that domain object until the user asks for a clarification of that expression. It then switches to using descriptive expressions for that domain object thereafter. Please note that users do not ask for any clarifications when descriptive expressions are used. This policy follows the same strategy for every domain object mentioned in the conversation. It therefore adapts only after a clarification is asked by the user for each jargon referring expression. We called this kind of adaptation *local adaptation* in chapter 7. This policy performs well with expert users (AA = 96%) but very poorly with novice users (AA = 47%) based on simulated user evaluation. An example dialogue between the dialogue system using Jargon-adapt policy and a real user is given in Appendix D.

8.1.2 Learned DS policy

The Learned-DS policy was the policy trained using our data-driven class-based three-step user simulation. As discussed in chapter 7, this policy learned from interacting with three groups of users (novices, intermediate-2, and experts). It learned how to identify the members of each group and adapt to them effectively. It also learned to generalise what it learned to users it was never exposed to during the training phase (i.e. intermediate-1 and intermediate-3 users). We found that this policy performs very well with all five user groups during simulated user evaluation with an overall average adaptation accuracy of 79.7%.

The decision to choose between using jargon expressions and descriptive expressions for the referents is made based on the Q-values of the two actions (i.e. *choose_jargon* and *choose_desc*) in the given user model state. The Q-value of each action (a) is calculated using the following formula:

$$Q(s,a) = \sum_{i=1}^{n} \theta_a(i)s(i)^T$$

where, s is the user model state with n variables

As explained in section 7.1.2, there are 26 variables (i.e. $n = 26$) in the user model s (s^T is the transpose of s). These variables are initially set to 0, signifying that the agent does not know what the user's domain knowledge levels are. These are then set or reset to 1 or 0, based on the evidence available during the conversation. For each action a, the NLG agent learns θ values for each of these variables in the user model ($\theta_a = \theta_a(1), \theta_a(2), .., \theta_a(n)$), during the training phase. Therefore, for each referent, the agent learns 2 sets of θ values, one for each action. The θ values signify the relevance of user's knowledge of various jargon expressions in the domain to its actions. The action that gets the highest Q-value is executed.

8.2 Evaluation setup

8.2.1 Wizard Interface Tool

For evaluation with real users, the Wizard-of-Oz setup that was used for data collection was reused (Figure 8.1). The wizard interface tool was re-designed to suit the needs of the evaluation process as shown in the figure 8.2. The following changes were made to the tool that we used for data collection:

1. All different kinds of clarification requests from the user were merged and categorised as the request_clarification dialogue act instead of the different dialogue acts used in the data collection setup.

2. The user's choice of referring expressions that were captured by the User's RE Choice panel was removed because we could not study lexical alignment using the data in our corpus.

3. Similarly, the capacity to align with users lexically was also removed.

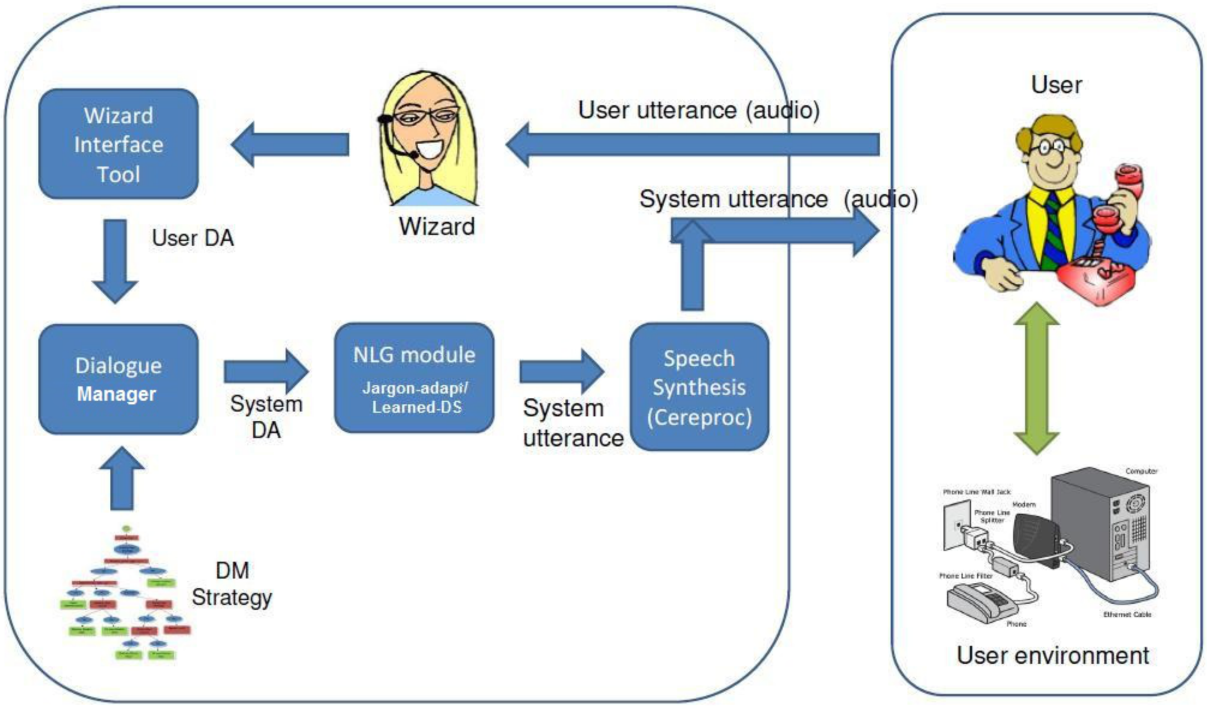
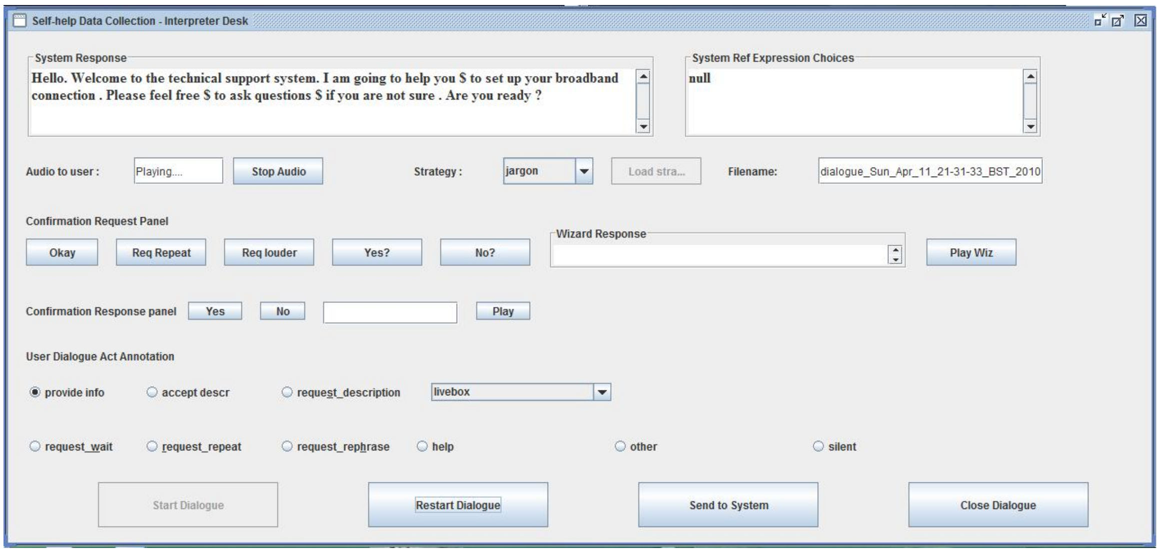


Figure 8.1: Wizard-of-Oz setup - Evaluation


Self-help Data Collection - Interpreter Desk

System Response: Hello. Welcome to the technical support system. I am going to help you $ to set up your broadband connection . Please feel free $ to ask questions $ if you are not sure . Are you ready ?

System Ref Expression Choices: null

Audio to user : Playing....

Strategy : jargon

Filename: dialogue_Sun_Apr_11_21-31-33_BST_2010

Confirmation Request Panel

Wizard Response

Confirmation Response panel

User Dialogue Act Annotation

(● provide info) accept descr request_description livebox

request_wait request_repeat request_rephrase help other silent

Figure 8.2: Wizard Interaction Tool - Evaluation

8.2.2 NLG module

The NLG module can be set to two different strategies - Learned-DS or Jargon-adapt. The strategies select between Jargon and Descriptive referring expressions for the domain entities used in the system utterances. The NLG module automatically replaces the RE slots with the expressions chosen by the strategies and the final utterances are then generated.

8.2.3 User's environment

The environment was modified to be more challenging than the one used in the data collection exercise. Recall that we trained the system to learn a user modelling policy for a modified environment. Therefore these modifications were made to keep the evaluation environment consistent with the environment provided for training the policy (see section 7.2.1). The following modifications were made to the evaluation environment that the participants interacted with:

1. Text labels like "ADSL", "Ethernet", etc for sockets on the Livebox were deleted by whitening them out. However, symbolic labels on the status lights were not altered.

2. One of the two broadband filters was replaced with a telephone-pin converter (distractor).

3. A USB cable (distractor) was added to the environment along with the two other cables already supplied.

Similarly, contextual clues in the instructions were removed. For example, when broadband filters were first mentioned, the additional clue (i.e., "there are two of them in the package") has been removed. Figure 8.3 shows the environment used for evaluation. It shows how the environment has been modified to make the process of identifying domain objects more difficult than it was during our data collection exercise (see figure 6.3).

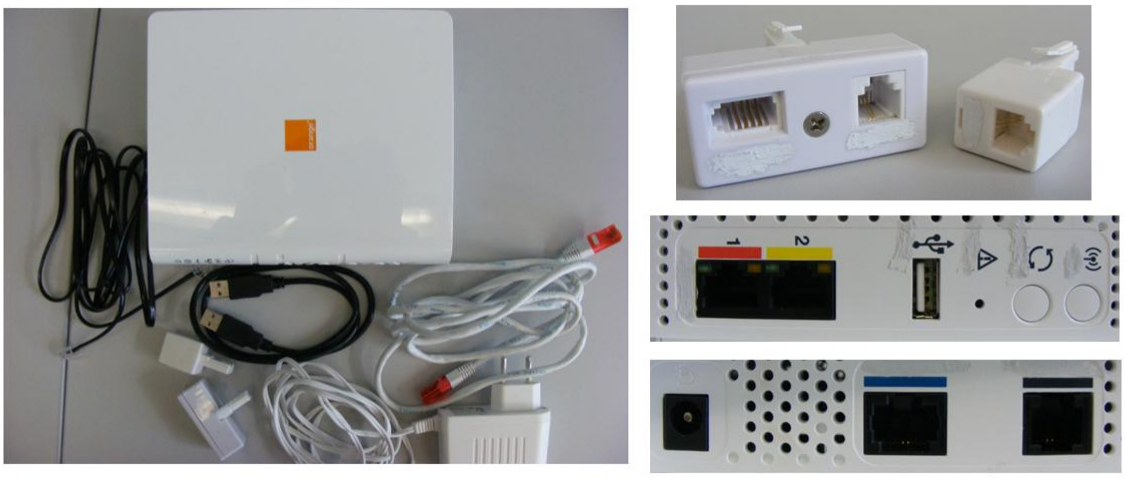


Figure 8.3: Objects in user environment during evaluation

8.3 Evaluation process

We followed a step-by-step process to collect all necessary data from the participant:

1. Background questionnaire: The participant's background information including their profession, gender, age group, and their previous experience in setting up broadband Internet connections and in using spoken dialogue systems were collected (see Appendix C.2).

2. Pre-task recognition test: A list of 13 technical terms were read out aloud to the participants. They were asked to point to the domain objects based on their knowledge of the domain and educated guesses (see Appendix C.4).

3. Dialogue task: Each participant was allowed to do the dialogue task only once. Therefore, for each participant one of the two strategies were randomly chosen. The participants then interacted with the dialogue system using headphones and a wireless mic. The system gave them step-by-step instructions to setup the broadband connection using the objects in front of them in their environment. The participants responded to the instructions with acknowledgements, environmental status information, requests for clarification, and other dialogue actions.

4. Post-task recognition test: The participants were given the same recognition test as the pre-task recognition test to collect their domain knowledge after the dialogue task.

5. Task completion rate: The experimenter then examined the broadband setup to calculate the task completion rate. This was revealed to the participant (see Appendix C.5).

6. User feedback: The participants filled out a user feedback questionnaire on the different features of the system based on the conversation (see Appendix C.3).

7. Debriefing: The participants were informed about the real setup and the role of the wizard in the setup.

Please note that the questionnaires used for evaluation were slightly different from the ones used for corpus collection. In the evaluation questionnaires, questions about the system's adaptive features were asked. These questions did not figure in the data collection questionnaire because the strategies used then were not adaptive.

8.4 Data

The two strategies were evaluated on 36 participants. All of the participants were students, but they came from different schools (arts, humanities, science and engineering). Besides the forms filled in by the participant and the experimenter, the interaction between the system and the participant was audio recorded and logged [1]. Objective parameters like dialogue time (DT), turn count, number of jargon and descriptive terms, number of clarification requests, etc was automatically logged along with the conversation. Normalised learning Gain (LG) was calculated using the pre-test and post-test

[1]This data will be made available as part of the CLASSiC project (www.classic-project.org) deliverables

recognition task scores. 17 participants (Group JA) used a dialogue system that executed the Jargon-adapt policy and 19 others (Group LDS) used a system that executed the Learned-DS policy. The initial knowledge of the users (mean pre-task recognition score) of the two groups were not significantly different from each other (JA = 7.35 $\pm$ 1.9, LDS = 6.57 $\pm$ 2.29). Hence there is no bias on the user's pre-task score towards any strategy.

Technical referring expressions like "Livebox", "Phone socket", etc that belong to the *easy* class were known to almost all participants. However, expressions like "Broadband filter" and "Livebox ethernet socket" were known only to a few users. The knowledge frequency of each technical expression is given in figure 8.4. This data, as expected, is consistent with the changes we made to the environment before we evaluated the strategies. For example, fewer participants identified the ADSL socket in the evaluation setup than in the data collection setup.

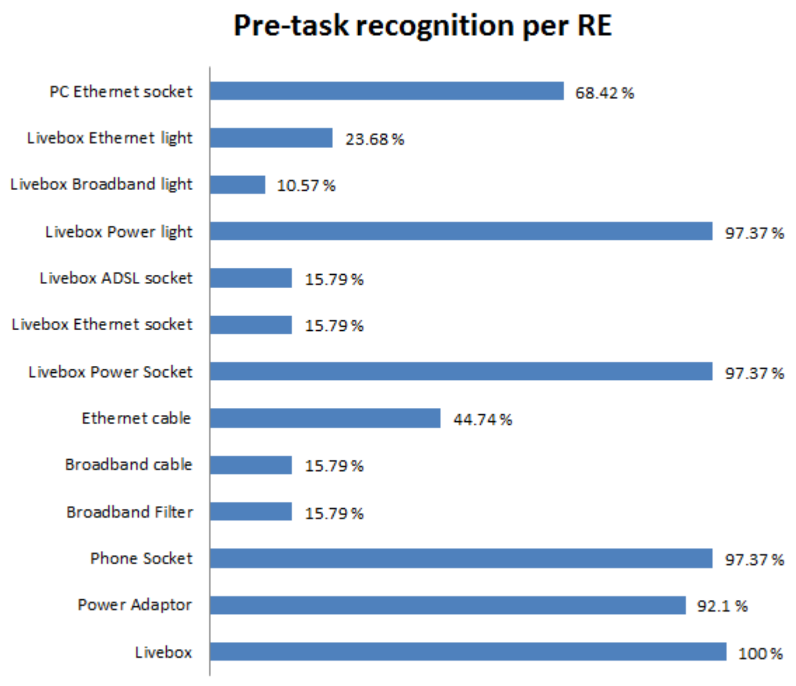


Figure 8.4: Pre-task recognition frequency per RE

Similarly, we also clustered the initial knowledge profiles of the participants using their pre-task scores into 5 clusters. Fig 8.5 shows the frequency of each cluster. The centroid profiles of each cluster were the same as the knowledge profiles we used in our data-driven user simulations. Please note that this pre-task test was carried out in our new challenging environment. This validates our manual modification described in chapter 6 that was done keeping in mind the disparity between the environment used in the data collection and the one we use currently in real user evaluation.

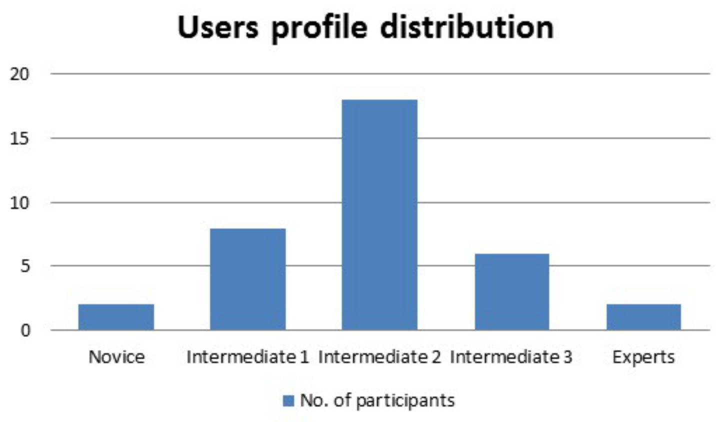


Figure 8.5: User pre-task knowledge profile distribution

8.5 Results

Table 8.1 presents the mean Adaptation Accuracy (AA), Dialogue Time (DT), Task Completion Rate (TCR), (normalised) Learning Gain (LG), etc produced by the two strategies that were evaluated with real users. Tests for statistical significance were done using the two-tailed Mann-Whitney test for 2 independent samples (due to the non-parametric nature of the data).

The Learned-DS strategy produced more accurate adaptation than the Jargon-adapt strategy. The difference between the means was statistically significant ($p = 0.000$). It also had significantly higher task completion rate (TCR) than the Jargon-adapt policy ($p = 0.000$). The Learned-DS strategy produced significantly lower dialogue time (DT) than the Jargon-adapt policy ($p = 0.005$) and fewer turns ($p = 0.012$). The Learned-DS strategy saved time by almost 11.05% when compared to the Jargon-adapt strategy. This time saving is due to the fact that users are able to find the intended referrent faster when the system adapted to the user. The Learned-DS strategy produced less jargon according to the needs of the user and therefore elicits fewer clarification requests (CR). Results therefore show that Learned-DS strategy is significantly better than the hand-coded Jargon-adapt policy in terms of adaptation accuracy, dialogue time and task completion rate. There was no difference between the performance of both the strategies in terms of (normalised) learning gain. However, this is not a problem as our objective was not to increase learning gain.

Groups	JA	LDS
Strategy used	Jargon-adapt	Learned-DS
Adaptation Accuracy (%)	63.91 (± 8.4)	84.72***(± 4.72)
Learning Gain	0.71 (± 0.26)	0.74 (± 0.22)
Dialogue Time (mins)	7.86 (± 0.77)	6.98**(± 0.93)
Task Completion Rate (%)	84.7 (± 14.63)	99.47***(± 2.29)
No. of Turns	30.05 (± 2.33)	28.10*(± 1.85)
No. of Clarification Req	4.11 (± 2.36)	2.79 (± 1.43)

* Statistical significance ($p < 0.05$).

** Statistical significance ($p < 0.001$).

*** Statistical significance ($p < 0.0001$).

Table 8.1: Evaluation with real users

8.5.1 Low vs High users

The participants can be analysed based on their pre-task scores into two subgroups (Low and High). Low users are those whose pre-task scores are lesser than the mean pre-task score and High users are those whose pre-task scores are higher. We made such a division for the users of each group (JA and LDS) to examine the effects of adaptation on users with different levels of domain expertise. Figures 8.7, 8.6 and 8.8 show how the two different systems performed with the two groups of users.

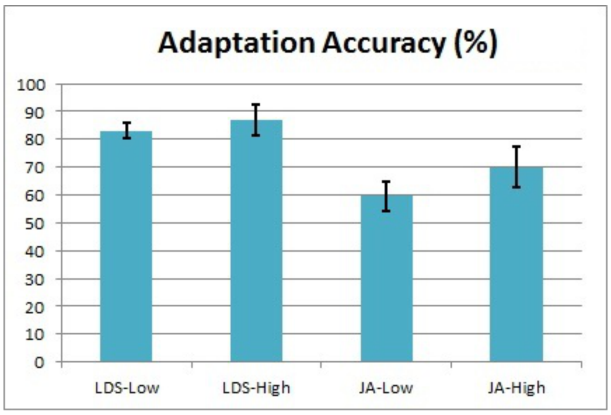


Figure 8.6: Accuracy vs User Expertise

Accuracy of the Learned-DS system is not very different for both the user groups: Low and High. For Low group users, the accuracy is 83% (±2.72) whereas for High group users it is 86.98% (±5.57). The reason why Low group users have lower accuracy because the system has to spend a few extra turns sensing the domain knowledge of the user. The Jargon-adapt system does not perform well with the Low group users (59.63%) whereas it performs better with High group users (70%).

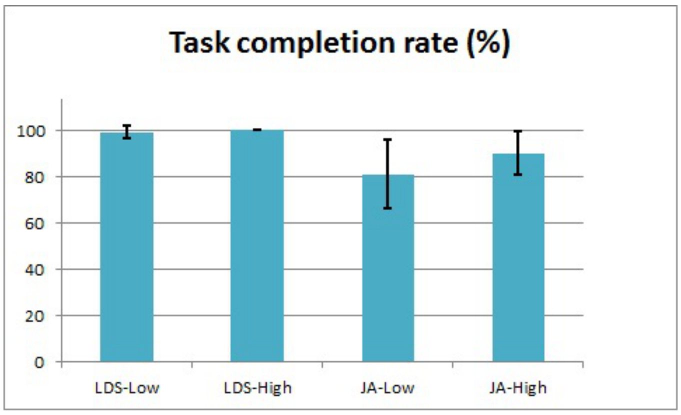


Figure 8.7: Task Completion Rate vs User Expertise

Fig 8.7 shows how the two policies adapt to users having different levels of domain knowledge before the dialogue task. The Jargon-adapt policy adapts very well to High users whereas it performs poorly with Low users. Its performance increases with users' expertise. Since the Jargon-adapt policy is designed to use jargon expressions in the first instance, it performs well with expert users. On the other hand, the Learned-DS policy adapts consistently with all users across the knowledge spectrum. By choosing the appropriate expression based on user's expertise, it seems make following the instructions easier and therefore gives a higher overall task completion rate (99.47%).

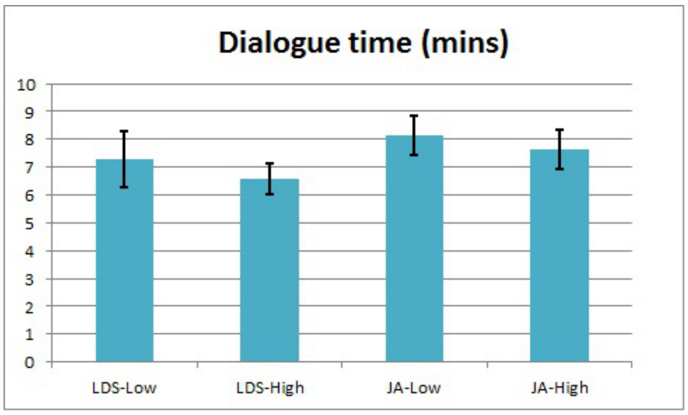


Figure 8.8: Dialogue Time vs User Expertise

In terms of dialogue time, the Learned-DS system consistently produces lower dialogue time for both the groups than the Jargon-adapt system. Especially, the High

Groups	JA	LDS
Strategy used	Jargon-adapt	Learned-DS
Q1. Quality of voice	3.11	3.36
Q2. Had to ask too many questions	2.23	1.89
Q3. System adapted very well	3.41	3.58
Q4. Easy to identify objects	2.94	3.42
Q5. Right amount of dialogue time	3.23	3.26
Q6. Learned useful terms	2.94	3.05
Q7. Conversation was easy	3.17	3.42
Q8. Future use	3.23	3.47

Table 8.2: Real user feedback

group users seemed to save almost a minute when they used the Learned-DS system compared with their counterparts using the Jargon-adapt system.

8.6 User satisfaction scores

Table 8.2 presents how users subjectively scored different features of the system based on their conversations with the two different strategies. Statistical significance in difference between the means were also tested using the two-tailed Mann-Whitney test for 2 independent samples.

User's feedback on different features of the systems were not very different from each other. The Learned-DS policy was rated slightly higher than its counterpart on all questions. However, the differences were not statistically significant. Users of the system employing the Learned-DS policy found it easier to identify domain objects (Q4) during the interaction ($p = 0.043$). However, when Bonferroni correction is applied (i.e. $\alpha = 0.006$), it is not statistically significant.

Overall, the differences in the objective parameters like task completion rate, dialogue time, etc were not supported by the user's feedback. Users seemed to not be able to recognize the nuances in the way the system adapted to them. This is evident from user ratings for Q3 (i.e. System adapted very well). They could have been satisfied with the fact that the system adapted (locally) at all. This adaptation and the fact that the system offered help when the users were confused in interpreting the technical terms, could have led the users to score both systems well in terms of dialogue time (Q5), ease of conversation (Q7) and future use (Q8). The users were given only one of the two strategies and therefore were not in a position to compare the two strategies and judge which one is better. The differences in user score (and the objective parameters) were therefore only compared using Mann-Whitney U test for independent samples. Please note that, as pointed out (in chapter 7), user scores were not discriminative enough to be used as a reward function. Results in table 8.2 lead us to conclude that perhaps users need more information like experiencing two or more strategies in order to judge the strategies better. It is not clear how well the users understand the nuances in natural language generation of the system and therefore whether it is a good idea to use user scores for training and evaluation.

We tested for correlation between the above parameters using two-tailed Spearman's Rho correlation. We also found that accuracy of adaptation (AA) correlates positively with task completion rate (TCR) ($r = 0.584, p = 0.000$) and negatively with dialogue time (DT) ($r = -0.546, p = 0.001$). However, there are no correlations between these parameters when the two groups (i.e. Jargon-adapt and Learned-DS) are investigated separately. These correlations and our results suggest that as a system's adaptation towards its users increases, the task completion rate increases and dialogue duration decreases significantly.

8.7 Real vs Simulated evaluation

Our results show that the results of evaluation with simulated users transfer to real users as well. In our simulated evaluation (see chapter 7), we found that the Jargon-Adapt policy adapted to different users less than the Learned-DS policy. We found a similar trend in the real user evaluation. The results in real user evaluation show that the adaptation accuracy is much lower for Jargon-Adapt and much higher for Learned-DS than the simulated evaluation. This is because, in simulated evaluation, all user types were equally distributed, whereas in the real user evaluation, there were more intermediates than novices and experts (see figure 8.5). We also find the same trend in terms of dialogue time. The Learned-DS policy produced shorter dialogues than the Jargon-adapt policy in both evaluations. However, there is a difference due to the fact that the Internet setting-up task done by the participants in the real evaluation was more difficult than the one used in the data collection from which the linear model for calculating dialogue time in simulation was derived. Our results therefore reinforce such trends shown in previous studies that used reinforcement learning techniques for dialogue policies (Henderson et al. (2005, 2008); Rieser (2008)).

8.8 Qualitative analysis

Table 8.3 presents samples of a dialogue between a real user (from the above evaluation) and the dialogue system using the Learned-DS policy. The dialogue shows how the user asks for clarification on two jargon expressions: "broadband cable" and "broadband filter" and how the system uses descriptive expressions to refer to broadband filter ("small white box..") and broadband light ("second light..") later on in the dialogue. Please note that there are two kinds of adaptations here. The system clearly knows that the user had asked for a clarification on the jargon expression "broadband filter" and therefore does not know what it means. It therefore switches to its corresponding descriptive expression. This is an example of *local* adaptation. On the other hand, it also chose to use a descriptive expression for the broadband light. This is due to the system's ability to predict the user's knowledge of the jargon expression "broadband light" based on the information it already gathered about the user's domain knowledge. This is an example of global adaptation. The system used jargon for broadband filter in the first instance because it might not have had enough data for predicting the user's unknown domain knowledge. (See Appendix D for a transcript of the full conversation).

While the Learned-DS system adapts well to its users globally, the Jargon-adapt system adapts only locally. The Learned-DS strategy enabled the system to adapt using the dependencies between knowledge concepts and different knowledge patterns in users that it learned during the training phase. For instance, when the user asked for clarification on "ethernet cable", it used descriptive expressions for domain objects like ethernet light and ethernet socket on the Livebox. Such adaptation across referents enabled the Learned-DS system to score better than the Jargon-adapt strategy. On the other hand, the Jargon-adapt policy only adapts after learning completely about the user's knowledge and does not predict at all. Therefore, it performs worse than the Learned-DS policy.

Sys: Do you have a **broadband cable** in the package?

Usr: Which one is the broadband cable?

Sys: The broadband cable is the thin black cable with colourless plastic ends.

Usr: Yes.

....

Sys: Do you have a **broadband filter** in the package?

Usr: Which one is the broadband filter?

Sys: The broadband filter is the small white box that
has two sockets and an phone plug.

Usr: Yes.

....

Sys: Please plug the **small white box that has two sockets and a phone plug**
into the **phone socket** on the wall.

Usr: Yes.

....

Sys: What is the status of the **second light from the left** on the **Livebox**?

Usr: Its flashing red.

....

Table 8.3: Learned-DS policy - Dialogue with real user

8.9 Conclusion

In this chapter, we presented the evaluation framework that we employed to evaluate the two high-scoring strategies from the simulated user evaluation (see chapter 7). We presented the results of the evaluation of the strategies with real users, which showed that the Learned-DS strategy that was learned using the reinforcement learning framework and the data-driven user simulations adapted better to real users than the hand-coded Jargon-adapt strategy. We also showed that adaptation to the user's domain knowledge results in improved task success (99.47% for a learned policy vs 84.7% for a hand-coded adaptive policy) and reduced dialogue duration (11%). We also presented an analysis of the results that showed why the Learned-DS strategy scored well on objective parameters like dialogue time, accuracy of adaptation and task completion rates, and why most of the user satisfaction scores on the two different strategies are inconclusive.

Chapter 9

Conclusion

User-adaptive spoken dialogue systems have been an important research topic for researchers in Human-Computer Interaction. There are several aspects of users that an interactive system can adapt to. Some of these aspects include users' speech patterns, goals, preferences, system skills, environmental constraints, domain knowledge level, etc (Komatani and Okuno (2010); Chu-Carroll and Nickerson (2000a); Jokinen (2006); Komatani et al. (2005); Rogers et al. (2000); Carenini and Moore (2001); Walker et al. (2004); Stoia et al. (2006); Mairesse and Walker (2010); McKeown et al. (1993)). As seen in Chapter 1, adapting to users' domain knowledge levels has been shown to be very important in technical domains where users talk to systems for technical assistance, troubleshooting advice, etc. This involves adapting at various levels including the choice of referring expressions, and complexity of instructions. In this thesis we focussed on adaptation to users' domain knowledge at the level of referring expressions. Our objective was to build a dialogue system that can adapt to the user's capability to identify the domain objects using jargon expressions.

Another important dimension to this problem is how much information systems must have about users' domain knowledge in order to adapt to them. Many interactive systems in general and referring expression generation and natural language generation systems in particular have used static user models and therefore assumed that the systems are informed of the domain knowledge of the user beforehand (Dale (1989a); Reiter and Dale (1992, 1995); McKeown et al. (1993)). In contrast, this thesis addressed the idea of dynamic user modelling in which the system has no information as to what the user's domain knowledge levels are before the conversation starts (Rich (1979); Chin (1989); Cawsey (1993); Carberry et al. (1999)). Therefore, the challenge was to identify the domain expertise of the user dynamically during the conversation and adapt to him/her at the same time. This introduced an interesting trade-off between when to sense the user's knowledge and when to adapt to the user provided the modalities for sensing and adaptation are the same.

Several solutions to these problems have been proposed. Some systems used hand-coded rules to seek information about the user's domain knowledge (Rich (1979); Chin (1989); Cawsey (1993)). They used hand-coded rules that can infer unknown facts about the user's domain knowledge from what is known already. Other rules informed the system when and how to seek information about the state of user's knowledge when inference cannot be made using existing information. However, such approaches require considerable domain expertise during system development. Others used supervised learning approaches to address the problem of predicting the user's knowledge (Akiba and Tanaka (1994); Jameson (1995)). However, this did not automatically address the problem of when and how to sense information when prediction is impossible or unreliable. Besides, supervised learning approaches need large amounts of data to learn from. In contrast to these approaches, this thesis presented an alternative approach that addresses the twin problem of seeking information and predicting the user's knowledge for adaptation, using only small amounts of WoZ data. Our system learns to choose between two kinds of referring expressions: jargon and descriptive to refer to the domain entities. It learns to sense the initial state of user's domain knowledge and adapt to it during the course of the conversation.

9.1 Thesis contributions

RL framework for UM policy learning: This thesis presented a reinforcement learning framework that learns to model users with different levels of domain knowledge and adapt to them dynamically. We present the user modelling problem as a Markov Decision Process and use reinforcement learning algorithms to learn user modelling policies that adapt to users whose domain knowledge is not already known to the system. The agent learns to sense information from users unobtrusively and adapt to them. The agent also learns to find a beneficial trade-off between sensing and adaptation.

Novel user simulation models: This thesis also presented three-step pipeline models for user simulation that simulate the dialogue behaviour of real users in technical conversations. To our knowledge, this is the first user simulation that has been built to train dialogue systems to adapt at the level of referring expressions because unlike the previous user simulation models, these models simulate a variety of users with different levels of domain expertise and are sensitive to the referring expressions used by systems. We have also shown that in comparison to other types of models such as bigram, trigram an equal probabilty models, the three-step models are better at simulating real users' dialogue behaviour. We also showed that the policies learned using our simulation models adapted better than hand-coded policies and policies learned using a hand-coded simulation model.

Learning from limited resources: We showed that it is possible to populate statistical user simulation models using a small corpus of non-adaptive dialogues (i.e., a system that used only one kind of referring expression - jargon or descriptive throughout the conversation) and knowledge profiles of different user types, which can then be used for learning adaptive policies. This saves us from hand-coding an adaptive NLG module or employing a domain expert to play the role of an adaptive NLG module in a dialogue system used for data collection.

Learning a UM policy and evaluation with simulated users: This thesis experimentally verified the reinforcement learning framework in which we trained and evaluated a user modelling policy for adaptive referring expression generation on both simulated and real users. Using a data-driven three-step pipeline user simulation model, we trained the learning agent to learn a UM policy. The learned policy modelled unknown users by dynamically seeking information about their domain expertise and adapted to them by predicting unknown facts about their domain knowledge. We evaluated the learned policy with simulated users and showed that the learned policy adapted more accurately than other adaptive hand-coded policies. We also showed that the learned policy was able to generalise to user types it never interacted with during its training phase.

Performance of the learned policy on real users: In this study, we also evaluated the same learned policy with real users in a wizarded dialogue system setup in a real technical environment and showed that the results from simulated user evaluation transfer to real user evaluation as well. In fact, the learned policy adapted significantly better to users than a high-scoring hand-coded adaptive policy. We showed that the learned policy produced more adaptation (approx. 20%) than the Jargon-adapt policy. This is because the learned policy senses continuously and adapts to users taking into account the stochastic nature of their dialogue behaviour.

Effect of adaptation on real users: We also showed that using the results of our evaluation and correlation between adaptation accuracy, task completion rate and dialogue time that by improving adaptation accuracy at the level of user's knowledge of referring expression generations, their task success rates can be increased and overall dialogue time can be reduced.

9.2 Future applications

The framework in this study can be used in future task oriented spoken dialogue systems where adaptation to user's domain knowledge is important. The following are a few examples to show how widely this method can be applied:

1. An assistive health care system which interacts with patients to educate and assist them in taking care of themselves (Bickmore and Giorgino (2004)). Such a system should be able to adapt to patients' initial level of knowledge and in subsequent dialogues change its language according to the improvement in the patient's understanding and improving knowledge of the domain.

2. A city navigation system that interacts with locals and tourists (Rogers et al. (2000)). Such a system should use proper names and descriptions of landmarks appropriately to different users to guide them around the city.

3. A technical support system helping expert and novice users (Boye (2007)). Systems like the one described in this thesis can be employed in different technical domains like troubleshooting laptops, user manuals for complex gadgets, etc. They should use referring expressions appropriate to the user's expertise.

4. An Ambient Intelligence Environment in a public space (e.g., museum) interacting with visitors (Lopez-Cozar et al. (2005)). Such systems can guide visitors and describe the exhibits in a language that the user would appreciate and understand.

5. A tutorial dialogue system that tutors students or trains personnel in industry (Dzikovska et al. (2007)). Such systems should adapt to the needs of the learner in terms of their levels of understanding and expertise.

In all of the above examples, the task domain may contain a considerable number of domain entities that are addressed differently by different groups of people. In such cases, it is ideal that the system adapts to different users by dynamically modelling them during the conversation. However, one should also pay attention to the possible misunderstanding between the partners during such adaptation. For example, patients and doctors may use the same jargon words but may mean different things. Several studies have found that patients misunderstand medical terminology (Spiro and Heidrich (1983); Thompson and Pledger (1993); Lerner et al. (2000); Koch-Weser et al. (2009)). Such misunderstanding is also related to ethnicity. For instance, non-native English speakers have been shown to have low understanding of the term "unconscious" (Cooke et al. (2000)).

Therefore, the use of jargon words cannot be assumed to imply accurate knowledge of concepts. Users may have misconceptions which will affect the task supported by the dialogue. In this study, the user's knowledge of jargon expressions is simply represented using overlay models. However, in future, we will aim to represent users' misconceptions as well. Even in this study, users have misconceptions. This is reflected in their task completion rates. When users misunderstand the jargon expressions, their task completion is less successful. Imagine such scenarios in a system that provides health advice to patients. We certainly should pay attention to the fact that sometimes users have misconceptions about the domain entities and that their mere use of jargon expressions should not be construed as accurate knowledge of domain entities and concepts. Similarly, it is reported that patients do not question or ask for clarifications on unknown jargon expressions when doctors use them unintroduced in their conversation with their patients (Koch-Weser et al. (2009)). In this study, we have assumed that the user is able to identify the referent when he doesn't question the use of a jargon expression. However, in light of these studies in health care, we cannot always make such assumptions. Therefore user responses and use of jargon should be carefully observed and the uncertainty in their understanding should be modelled. Advanced modelling techniques that such as Partially Observable Markov Decision Processes could be used to model such uncertainties (Lemon et al. (2010)).

9.3 Future research directions

We believe that we have taken a first step in using reinforcement learning to drive user modelling in dialogue contexts. However, we believe that this framework could be extended systematically to address other issues in the development of dialogue systems. The following are some open questions that can be studied in future:

9.3.1 Levels of adaptation

Can this framework be extended to other levels of adaptation in dialogue systems? As discussed previously in chapter 1, a truly adaptive system must be able to adapt to the user's expertise at different levels of dialogue management and utterance generation. We believe that this framework can be extended systematically to dialogue management decisions such as complexity of instructions (Dale (1989a,b)), need for preparatory propositions (Arts (2004)), repetitions, and to utterance generation decisions like content selection and text structure besides referring expressions (Dethlefs and Cuayahuitl (2010)).

Such extensions require advanced user modelling of users' domain knowledge as against just their domain communication knowledge (Rambow (1990); Kittredge et al. (1991)). Users' capabilities of reasoning with domain entities have to be represented using a richer representation of the user's domain knowledge like first-order logic (Boutilier (2001); Kersting and De Raedt (2003); van Otterlo (2004)). User simulation models for such tasks will need additional steps in the pipeline to review the concepts presented by the system. Finally, such extensions require dialogue data that can be used to train simulation models that can respond to instructions at different levels of complexity. We hypothesize that parameters like dialogue time would be reduced greatly if the system adapted at all levels to the expertise of the users. Similarly, user simulation models could be extended to include retention models for the user's memory defining how long users retain new technical information. This would be useful in longer dialogue tasks where the same technical information needs to be recalled in different dialogue episodes.

9.3.2 Adapting to other user traits

Can this framework be used to learn adaptive policies to handle other user traits such as goals, personality, and emotions? As discussed in chapter 2, systems adapt to several user traits such as goals, preferences, system skills, and personality in addition to domain knowledge. The application of our reinforcement learning framework can be examined in dynamic adaptation to other user traits. This might require extending the user simulation models to include important features like the user's frustration, certainty in user response, goals, preferences, etc. (Tetreault and Litman (2006)).

9.3.3 Lexical alignment due to priming

Can this framework be extended to include lexical alignment through priming as well as adaptation? Lexical alignment in dialogue due to priming is the dialogue behaviour of using the same referring expressions or other lexical items as the interlocutor in the same or similar contexts (Pickering and Garrod (2004); Brennan (1991); Stenchikova and Stent (2007); Branigan et al. (2010)). This is also called *entrainment* (Porzel et al. (2006); Buschmeier et al. (2010)). For instance, in a conversation, an interlocutor who uses "chair" to refer to an object following the use of the same word by his partner, although he himself used the word "deckchair" before for the same referent, is said to be lexically aligning with his partner due to priming. In this thesis, we do not study this behaviour. We hypothesize that our framework could be extended to learn REG policies that could align lexically due to priming. Lexical alignment in dialogue systems have been studied previously by Isard et al. (2006); Buschmeier et al. (2010). In addition to adapting to user's expertise, the system could be trained to also lexically entrain to the user's vocabulary in dialogue tasks in which users are more vocal (i.e., produce more language) than the one used in this thesis. Entraining capabilities such as these could affect common ground between interlocutors and therefore affect task success and user satisfaction. Although we aimed to study entrainment in this work, the data collected from real users showed that in instruction giving-following tasks, users are less vocal than they usually are in information seeking tasks. Therefore we were unable to study entrainment behaviour in the current setup.

9.3.4 Addressing the attribute-selection problem

Can this framework be extended to lower-level decision making in referring expression generation? Recently, van Deemter (2009b,a); Golland et al. (2010) propose the application of ideas from *Decision theory* and *Game theory* to the problem of generating referring expressions in which the choices are made during generation based on their utility. Our framework presented a new approach to choose referring expressions. We consider the choice problem that we presented in this thesis as a problem of choosing type-labels (i.e., categories) in referring expression generation in technical domains for users with different domain knowledge levels. This, we believe, can be extended to the attribute selection problem (Dale (1989a); Reiter and Dale (1997)) where attributes like color, size and relational attributes are selected in a reward (or utility) driven manner. The choice of attributes dynamically change with the choice of the type label. Deciding to use specific technical type-labels reduces the number of attributes for disambiguation of the target entity among its distractors. Therefore it is an interesting problem to extend the reinforcement learning framework to not only identify the type label but also the attributes whose relevance is dynamically changing. It would also be interesting to learn policies that will select fewer attributes for subsequent references of domain entities compared to those used in the initial references.

9.3.5 Using POMDP models

Can we increase adaptation by considering the user's domain knowledge as partially observable rather than fully observable and account for such uncertain observations? Partially Observable MDPs (POMDP) could be used to model the user's expertise in the dialogue state, instead of using completely observable Markov Decision Processes (MDP) (Young (2006); Williams et al. (2006)). In an MDP environment, the agent is assumed to be able to accurately observe its environment/user. However, in reality, such accurate observations are hardly possible due to speech recognition errors, noisy environments and so on. Hence the agent must maintain a probability distribution over all possible states based on the observations. Recently, the use of POMDP models have been studied for the purpose of dialogue management. Williams and Young (2007) used POMDP models for troubleshooting tasks like fixing a broken broadband connection. However, POMDPs have not yet been used for learning NLG or user modelling policies. Using POMDPs for learning user modelling policies for NLG will let the dialogue system work directly with the uncertainty of the knowledge of the user's expertise.

9.3.6 Comparison to resource intensive approaches

Would the reinforcement learning approach perform better than resource intensive approaches such as hand-coded rules and supervised learning methods? In this thesis, we have specifically addressed the issue of scarcity of resources that are available to develop adaptive dialogue systems and have therefore not built systems either using domain expertise to hand-code rules or using supervised learning for comparison. Our approach bootstraps from dialogues between real users and non-adaptive dialogue systems that were easy to build. Although we showed that learned policies are better at adaptation than some hand-coded policies, we do not claim that the policies learned using our reinforcement learning framework would adapt better than policies learned from human experts using supervised learning or rule-based policies hand-coded by domain and conversational experts. We also found that the Stereotype strategy that uses user profiles from the data collected was as good as the Jargon-adapt policy. Therefore, it would be interesting to compare its performance with real users to that of the learned policies in the future.

Earlier, Rieser (2008) showed that policies learned using reinforcement learning were better than the ones learned using supervised learning. One should also note that it is difficult to find domain experts who may also be comfortable adapting to users with different domain knowledge levels (Hinds (1999)). However to study this in the context of technical domain conversations, we require either a large corpus of adaptive dialogues between domain experts and a variety of users or an expert to hand-code adaptation rules for different configurations of the user model state. Nevertheless, we believe that our approach (being less resource intensive) is a worthy tool to add to the "toolkit" for dialogue system development.

9.4 Summary

To summarize, in this thesis, we have addressed the problem of dynamic user modelling at the level of referring expression generation in the context of technical support dialogues in resource scarce conditions. We formulated the user modelling problem as a Markov Decision Process and presented a reinforcement learning framework to learn user modelling policies for adaptive referring expression generation. We presented a data collection framework which we used to collect dialogue data and other essential information from real users. We designed and trained novel user simulation models that simulate the dialogue and physical behaviour of real users in situated technical tasks such as setting up a broadband Internet connection. Our novel design allowed us to train the user simulation using non-adaptive dialogues and then use those models in adaptive dialogues. We showed how to train a user modelling policy using data-driven user simulation models and test the learned policy with simulated users. We also evaluated the learned policy with real users and showed that learned policies performed better than some hand-coded adaptive policies. Our study also shows that by adapting to users' domain knowledge by choosing appropriate referring expressions, it is possible to increase task completion rate and decrease dialogue duration.

We believe that this novel approach can be applied for learning adaptive policies at many levels including dialogue management, text structuring, and attribute selection. We also hope that it would be applied to adapt to other user traits such as goals, preferences, and emotions in addition to user's domain knowledge levels in the future. Although we are not sure if our approach produces better adaptive strategies than supervised or hand-coded strategies by domain experts, we believe that our approach to developing adaptive REG modules in particular and adaptive dialogue systems in general is a welcome addition to the domain of dialogue systems development.

Appendix A

Instructions for setting up home broadband connection

The instructions for setting up home broadband connection as given by the dialogue system to users is divided into three episodes. These instructions are modified from the original version on the Orange (France Telecom) website.

1. Welcome/introduction
2. Check for necessary domain objects
3. Setting up instructions

In the following instructions, all domain objects are referred to using jargon expressions. However, they were modified according to the NLG/REG strategy used by the system.

A.1 Welcome message

First, the system introduces itself to the user. The following message is sent (in audio) to the user.

"Hello. Welcome to the technical support system. I am going to help you to set up your broadband connection. Please feel free to ask questions if you are not sure. Are you ready?"

A.2 Check for domain objects

The following 5 steps are sent to the users to check if they have all the domain objects required to carry out the instructions in the following episode.

1. Do you have a Livebox in the package?
2. Do you have a broadband cable in the package?
3. Do you have an ethernet cable in the package?
4. Do you have a power adaptor in the package?
5. Do you have a broadband filter in the package?

A.3 Setting up instructions

In the last episode, the system gives the users instructions to set up the connections. Some of these instructions request users to observe the status of the domain objects. Instead of presenting them as declarative statements they were posed as questions. For example, instead of stating "Now, the broadband light will turn on", we ask "What is the status of the broadband light" to make the conversation more interactive.

1. Disconnect the phone from the phone socket on the wall.
2. Take the power adaptor.
3. Plug the power adaptor into the two-pin mains power socket.
4. Connect the cable of the power adaptor firmly into the power socket of the Livebox. Observe the lights on the front panel of the Livebox.
5. Did all the lights on the front panel of the Livebox turn on and go off after a few seconds?
6. Place the Livebox with the Orange label facing up. What is the status of the power light on the Livebox?
7. Please plug one end of the broadband cable into the ADSL socket on the Livebox.
8. Take the broadband filter.

9. Plug the other end of the broadband cable into the broadband filter.
10. Plug the broadband filter into the phone socket on the wall.
11. What is the status of the broadband light on the Livebox?
12. Plug one end of the ethernet cable into the ethernet socket on the Livebox.
13. Connect the other end of the ethernet cable into the ethernet socket on the back panel of your computer.
14. What is the status of the ethernet light on the Livebox?
15. Connect the phone cable into the broadband filter that you plugged into the phone socket on the wall.
16. We have now finished setting up your broadband Internet connection.

A.4 Comments

Please note that since we did not have a live phone-line based Internet connection to further extend the task, the dialogue task was limited to the physical task of setting up connections between the various components.

Please also note that the above instructions were very simple (activity-wise) and that all users can easily carry them out when provided the appropriate referring expressions. However, some users might be able to understand more complex instructions. For example, a complex instruction like "Connect the Livebox to the computer using the ethernet cable." has been simplified into two steps (steps 12 and 13). The choice between using a simple and complex instruction is a dialogue management problem and has not been explored in this thesis. However, such a task would also require user modelling and adaptation techniques like the ones used in this thesis.

Appendix B

Data Collection Questionnaire

B.1 Instructions

You have applied for a broadband connection and have just received a package from the Internet company. Your task is to talk to the automated computer system designed to help the customers to setup the broadband Internet connection using the package. The computer will give you instructions to setup the connection. Please follow them carefully.
IMPORTANT: Please do not dismantle the setup after the conversation.
NOTE: Please speak naturally as you would normally speak to a human operator
1. Wear the headset in front of you.
2. Say Im ready when you are ready.

The computer will greet you with a welcome message when you are ready. If you have any questions, please ask the experimenter now.

B.2 Questionnaire - User background

Profession:
Department/School :
Course:
Gender: Male / Female
Age group (please tick):
(a)below 20 (b)21-30 (c)31-40 (d)41-50 (e)above 50 years
1. How long do you use computers per day? hours

2. I use the Internet (Tick all that apply)
(a) At work (b) At home (c) At Internet cafes (d) not at all

3. My estimated daily usage of the Internet is
(a) less than 1 hr (b) 1 hr to 3 hrs (c) 3 hrs and more (d) none

Please tick:
4 a. Have you set-up an internet connection at home on your own? Yes / No
4 b. If yes, is it from Orange? Yes / No
4 c. If yes, have you ever called technical support on the phone for help? Yes / No

5 a. Have you ever talked to automated customer support systems (with computers instead of humans operators) Yes / No
5 b. If yes, did you find the conversation easy? Yes / No
5 c. If yes, did you fix your problem using the automated system? Yes / No

B.3 Questionnaire - User feedback

In the following statements, we want you to rate the different features of the software. Please circle the one reaction that best describes the extent to which you agree or disagree with each statement.
1. I am confident that I have completed the task successfully.
[ ]Strongly disagree [ ]Disagree [ ]Agree [ ]Strongly Agree

2. The quality of the computer's voice was very good.
[ ]Strongly disagree [ ]Disagree [ ]Agree [ ]Strongly Agree

3. It was very easy to identify the objects the computer was referring to.
[ ]Strongly disagree [ ]Disagree [ ]Agree [ ]Strongly Agree

4. I learned useful new technical terms during the conversation
[ ]Strongly disagree [ ]Disagree [ ]Agree [ ]Strongly Agree

5. The information in the instructions was neither simple nor complex and just about

right.
[ ]Strongly disagree [ ]Disagree [ ]Agree [ ]Strongly Agree

6. The whole conversation was of the right length.
[ ]Strongly disagree [ ]Disagree [ ]Agree [ ]Strongly Agree

7. Overall, the conversation was very easy.
[ ]Strongly disagree [ ]Disagree [ ]Agree [ ]Strongly Agree

8. I would definitely use a similar speech system (for different tasks) again in future.
[ ]Strongly disagree [ ]Disagree [ ]Agree [ ]Strongly Agree

B.4 Recognition test

IMPORTANT: To be filled by the experimenter
Call out each of the following referring expressions to the participant. Mark if he correctly or incorrectly recognises the domain object.

a. Livebox [ ] Correct [ ] Incorrect
b. Power adaptor [ ] Correct [ ] Incorrect
c. Phone socket Correct [ ] Correct [ ] Incorrect
d. Broadband filter [ ] Correct [ ] Incorrect
e. Broadband cable [ ] Correct [ ] Incorrect
f. Ethernet cable [ ] Correct [ ] Incorrect
g. Livebox power socket [ ] Correct [ ] Incorrect
h. Livebox Ethernet socket [ ] Correct [ ] Incorrect
i. Livebox ADSL socket [ ] Correct [ ] Incorrect
j. Livebox power light [ ] Correct [ ] Incorrect
k. Livebox broadband light [ ] Correct [ ] Incorrect
l. Livebox Ethernet light [ ] Correct [ ] Incorrect
m. PC Ethernet socket [ ] Correct [ ] Incorrect

B.5 Task Completion Chart

IMPORTANT: To be filled by the experimenter

Observe the broadband setup made by the participant and record if he/she has done the following.

1. Phone cable unplugged from the phone socket (yes / no)
2. Power adaptor plugged in to mains (yes / no)
3. Power adaptor cable plugged in to Livebox power socket (yes / no)
4. Broadband cable plugged in to the Livebox ADSL socket (yes / no)
5. Broadband cable plugged in to the filter in the phone socket (yes / no)
6. Broadband filter plugged in to the phone socket (yes / no)
7. Ethernet cable plugged in to the Livebox Ethernet Socket (yes / no)
8. Ethernet cable plugged in to the PC Ethernet Socket (yes / no)
9. Phone cable plugged in to the filter in the phone socket (yes / no)
10. Other filter unused (yes / no)

Appendix C

Evaluation Questionnaire

C.1 Instructions

You have applied for a broadband connection and have just received a package from the Internet company. Your task is to talk to the automated computer system designed to help the customers to setup the broadband Internet connection using the package. The computer will give you instructions to setup the connection. Please follow them carefully.

IMPORTANT: Please do not dismantle the setup after the conversation.

NOTE: Please speak naturally as you would normally speak to a human operator

1. Wear the headset in front of you.
2. Say Im ready when you are ready.

The computer will greet you with a welcome message when you are ready. If you have any questions, please ask the experimenter now.

C.2 Questionnaire - User background

Profession:

Department/School :

Course:

Gender: Male / Female

Age group (please tick):

(a)below 20 (b)21-30 (c)31-40 (d)41-50 (e)above 50 years

1. How long do you use computers per day? hours

2. I use the Internet (Tick all that apply)
(a) At work (b) At home (c) At Internet cafes (d) not at all

3. My estimated daily usage of the Internet is
(a) less than 1 hr (b) 1 hr to 3 hrs (c) 3 hrs and more (d) none

Please tick:
4 a. Have you set-up an internet connection at home on your own? Yes / No
4 b. If yes, is it from Orange? Yes / No
4 c. If yes, have you ever called technical support on the phone for help? Yes / No

5 a. Have you ever talked to automated customer support systems (with computers instead of humans operators) Yes / No
5 b. If yes, did you find the conversation easy? Yes / No
5 c. If yes, did you fix your problem using the automated system? Yes / No

C.3 Questionnaire - User feedback

In the following statements, we want you to rate the different features of the software. Please circle the one reaction that best describes the extent to which you agree or disagree with each statement.
1. The quality of the computer's voice was very good.
[ ]Strongly disagree [ ]Disagree [ ]Agree [ ]Strongly Agree

2. I had to ask too many questions because I did not understand the words used by the system to refer to the different objects.
[ ]Strongly disagree [ ]Disagree [ ]Agree [ ]Strongly Agree

3. The computer adapted very well to my knowledge and used the right words to help me identify the objects.
[ ]Strongly disagree [ ]Disagree [ ]Agree [ ]Strongly Agree

4. It was very easy to identify the objects the computer was referring to.
[ ]Strongly disagree [ ]Disagree [ ]Agree [ ]Strongly Agree

5. The conversation took the right amount of time.
[ ]Strongly disagree [ ]Disagree [ ]Agree [ ]Strongly Agree

6. I learned useful new technical terms during the conversation
[ ]Strongly disagree [ ]Disagree [ ]Agree [ ]Strongly Agree

7. Overall, the conversation was very easy.
[ ]Strongly disagree [ ]Disagree [ ]Agree [ ]Strongly Agree

8. I would definitely use a similar speech system (for different tasks) again in future.
[ ]Strongly disagree [ ]Disagree [ ]Agree [ ]Strongly Agree

C.4 Recognition test

IMPORTANT: To be filled by the experimenter
Call out each of the following referring expressions to the participant. Mark if he correctly or incorrectly recognises the domain object.

a. Livebox [ ] Correct [ ] Incorrect
b. Power adaptor [ ] Correct [ ] Incorrect
c. Phone socket Correct [ ] Correct [ ] Incorrect
d. Broadband filter [ ] Correct [ ] Incorrect
e. Broadband cable [ ] Correct [ ] Incorrect
f. Ethernet cable [ ] Correct [ ] Incorrect
g. Livebox power socket [ ] Correct [ ] Incorrect
h. Livebox Ethernet socket [ ] Correct [ ] Incorrect
i. Livebox ADSL socket [ ] Correct [ ] Incorrect
j. Livebox power light [ ] Correct [ ] Incorrect
k. Livebox broadband light [ ] Correct [ ] Incorrect
l. Livebox Ethernet light [ ] Correct [ ] Incorrect
m. PC Ethernet socket [ ] Correct [ ] Incorrect

C.5 Task Completion Chart

IMPORTANT: To be filled by the experimenter

Observe the broadband setup made by the participant and record if he/she has done the following.

1. Phone cable unplugged from the phone socket (yes / no)
2. Power adaptor plugged in to mains (yes / no)
3. Power adaptor cable plugged in to Livebox power socket (yes / no)
4. Broadband cable plugged in to the Livebox ADSL socket (yes / no)
5. Broadband cable plugged in to the filter in the phone socket (yes / no)
6. Broadband filter plugged in to the phone socket (yes / no)
7. Ethernet cable plugged in to the Livebox Ethernet Socket (yes / no)
8. Ethernet cable plugged in to the PC Ethernet Socket (yes / no)
9. Phone cable plugged in to the filter in the phone socket (yes / no)
10. Other filter unused (yes / no)

Appendix D

Example dialogues

D.1 Example dialogue between the system using Learned-DS policy and a real user

Sys: Are you ready?

Usr: Yes.

Sys: Do you have a **livebox** in the package?

Usr: Yes.

Sys: Do you have a **broadband cable** in the package?

Usr: Which one is the broadband cable?

Sys: The broadband cable is the thin black cable with colourless plastic ends.

Usr: Yes.

Sys: Do you have an **ethernet cable** in the package?

Usr: Yes.

Sys: Do you have a **power adaptor** in the package?

Usr: Yes.

Sys: Do you have a **broadband filter** in the package?

Usr: Which one is the broadband filter?

Sys: The broadband filter is the small white box that has two sockets
and an phone plug.

Usr: Yes.

Sys: Good. We seem to have everything we need. Now follow
my instructions to set up the connections. Tell me when you are ready.

Usr: I am ready.

Sys: Please disconnect the phone from the **phone socket** on the wall.

Usr: Done.

Sys: Please take the **power adaptor**.

Usr: Yes.

Sys: Please plug the **power adaptor** into the two-pin mains power socket.

Usr: Yes.

Sys: Please connect the cable of the **power adaptor** firmly into the **power socket** of the **livebox**. Observe the lights on the front panel of the **livebox**.

Usr: Yes.

Sys: Did all the lights on the front panel of the **livebox** turn on and go off after a few seconds.

Usr: Yes.

Sys: Place the **livebox** with the Orange label facing up. What is the status of the **power light** on the **livebox**?

Usr: Its flashing red.

Sys: Please plug one end of the **thin black cable with colourless plastic ends** into the **ADSL socket** on the **livebox**.

Usr: Yes.

Sys: Take the **small white box that has two sockets and a phone plug**.

Usr: Yes.

Sys: Please plug the other end of the **thin black cable with colourless plastic ends** into the **small white box that has two sockets and a phone plug**.

Usr: Yes.

Sys: Please plug the **small white box that has two sockets and a phone plug** into the **phone socket** on the wall.

Usr: Yes.

Sys: What is the status of the **second light from the left** on the **livebox**?

Usr: Its flashing red.

Sys: Please plug one end of the **ethernet cable** into the **square socket with red stripe** on the **livebox**.

Usr: Yes.

Sys: Please connect the other end of the **ethernet cable** into the **ethernet socket** on the back panel of your computer.

Usr: Yes.

Sys: What is the status of the **fourth light from the left** on the **livebox**?

Usr: Its red.

Sys: Please connect the phone cable into the **small white box that has two sockets and a phone plug** that you plugged into the phone socket on the wall.

Usr: Ok.

Sys: Good. We have now finished setting up your broadband Internet connection.

D.2 Example dialogue between the system using Jargon-adapt policy and a real user

Sys: Are you ready?

Usr: Yes.

Sys: Do you have a **livebox** in the package?

Usr: Yes.

Sys: Do you have a **broadband cable** in the package?

Usr: Which one is the broadband cable?

Sys: The broadband cable is the thin black cable with colourless plastic ends.

Usr: Yes.

Sys: Do you have an **ethernet cable** in the package?

Usr: Yes.

Sys: Do you have a **power adaptor** in the package?

Usr: Yes.

Sys: Do you have a **broadband filter** in the package?

Usr: Which one is the broadband filter?

Sys: The broadband filter is the small white box that
has two sockets and an phone plug.

Usr: Yes.

Sys: Good. We seem to have everything we need. Now follow
my instructions to set up the connections. Tell me when you are ready.

Usr: I am ready.

Sys: Please disconnect the phone from the **phone socket** on the wall.

Usr: Done.

Sys: Please take the **power adaptor**.

Usr: Yes.

Sys: Please plug the **power adaptor** into the two-pin mains power socket.

Usr: Yes.

Sys: Please connect the cable of the **power adaptor** firmly into the **power socket** of the **livebox**. Observe the lights on the front panel of the **livebox**.

Usr: Yes.

Sys: Did all the lights on the front panel of the **livebox** turn on and go off after a few seconds.

Usr: Yes.

Sys: Place the **livebox** with the Orange label facing up. What is the status of the **power light** on the **livebox**?

Usr: Its flashing red.

Sys: Please plug one end of the **thin black cable with colourless plastic ends** into the **ADSL socket** on the **livebox**.

Usr: Yes.

Sys: Take the **small white box that has two sockets and a phone plug**.

Usr: Yes.

Sys: Please plug the other end of the **thin black cable with colourless plastic ends** into the **small white box that has two sockets and a phone plug**.

Usr: Yes.

Sys: Please plug the **small white box that has two sockets and a phone plug** into the **phone socket** on the wall.

Usr: Yes.

Sys: What is the status of the **broadband light** on the **livebox**?

Usr: Its flashing red.

Sys: Please plug one end of the **ethernet cable** into the **ethernet socket** on the **livebox**.

Usr: Yes.

Sys: Please connect the other end of the **ethernet cable** into the **ethernet socket** on the back panel of your computer.

Usr: Yes.

Sys: What is the status of the **ethernet light** on the **livebox**?

Usr: Its red.

Sys: Please connect the phone cable into the **small white box that has two sockets and a phone plug** that you plugged into the phone socket on the wall.

Usr: Ok.

Sys: Good. We have now finished setting up your broadband Internet connection.

Bibliography

Acomb, K., Bloom, J., Dayanidhi, K., Hunter, P., Krogh, P., Levin, E., and Pieraccini, R. (2007). Technical Support Dialog Systems: Issues, Problems, and Solutions. In *Proceedings of Bridging the Gap: Academic and Industrial Research in Dialog Technologies Workshop, NAACL-HLT, Rochester 2007*.

Ai, H. and Litman, D. (2007). Knowledge consistent user simulations for dialog systems. In *Proceedings of Interspeech 2007, Antwerp, Belgium*.

Ai, H. and Litman, D. (2009). Setting Up User Action Probabilities in User Simulations for Dialog System Development. In *Proceedings 47th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL), Singapore*.

Akiba, T. and Tanaka, H. (1994). A Bayesian approach for User Modelling in Dialogue Systems. In *Proceedings of the 15th conference on Computational Linguistics - Volume 2, Kyoto*.

Albrecht, D. W., Zukerman, I., and Nicholson, A. E. (1998). Bayesian Models for Keyhole Plan Recognition in an Adventure Game. *User Modeling and User-Adapted Interaction*, 8(1-2):5–47.

Alves, D. P., Weigang, L., and Souza, B. B. (2008). Reinforcement Learning to Support Meta-Level Control in Air Traffic Management. In Weber, C., Elshaw, M., and Mayer, N. M., editors, *Reinforcement Learning*. I-Tech Education and Publishing.

Androutsopoulos, I., Kokkinaki, V., Dimitromanolaki, A., Calder, J., Oberlander, J., and Not, E. (2001). Generating multilingual personalized descriptions of museum exhibits: the M-PIRO project. In *Proceedings of the International Conference on Computer Applications and Quantitative Methods in Archaeology. Gotland, Sweden*.

Appelt, D. (1985). *Planning English Referring Expressions*. Cambridge University Press, New York.

Araki, M. and Doshita, S. (1997). Automatic Evaluation Environment for Spoken Dialogue Systems. In Mayer, E., Mast, M., and Luperfoy, S., editors, *Dialogue Processing in Spoken Language Systems*, pages 183–194. Springer.

Arts, A. (2004). *Overspecification in Instructive Text*. Ph.D. thesis, Tilburg University, The Netherlands.

Balabanovic, M. (1998). Exploring versus Exploiting when Learning User Models for Text Recommendation. *User Modeling and User-Adapted Interaction*, 8:71–102.

Barto, A. G., Sutton, R. S., and Watkins, C. (1990). Learning and sequential decision making. In Gabriel, M. and Moore, J. W., editor, *Learning and Computational Neuroscience: Foundations of Adaptive Networks*, pages 539–602. MIT Press.

Barzilay, R. and Lapata, M. (2005). Collective Content Selection for Concent-to-text Generation. In *Proceedings of EMNLP'05.*

Basu, C., Hirsh, H., and Cohen, W. (1998). Recommendation as classification: Using social and content-based information in recommendation. In *In Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 714–720. AAAI Press.

Becker, T., Gerstenberger, C., Kruijff-Korbayova, I., Korthauer, A., Pinkal, M., Pitz, M., Poller, P., and Schehl (2006). Natural and intuitive multimodal dialogue for In-Car Applications: The SAMMIE System. In *Proceedings of 4th Prestigious Applications of Intelligent Systems (PAIS).*

Beebe, S. A. and Beebe, S. J. (2003). *Public Speaking: An Audience-Centered Approach.* Allyn & Bacon.

Bell, A. (1984). Language style as audience design. *Language in Society*, 13(2):145–204.

Bellman, R. (1957). A Markovian Decision Process. *Journal of Mathematics and Mechanics*, 6.

Belz, A. and Varges, S. (2007). Generation of Repeated References to Discourse Entities. In *Proceedings ENLG-2007.*

Bestavros, A. (1996). Speculative Data Dissemination and Service to Reduce Server Load Network Traffic and Service Time in Distributed Information Systems. In *Proceedings of the 1996 International Conference on Data Engineering.*

Bickmore, T. and Giorgino, T. (2004). Some Novel Aspects of Health Communication from a Dialogue Systems Perspective. In *AAAI Fall Symposium on Dialogue Systems for Health Communication.*

Boutilier, C. (2001). Planning and programming with first-order markov decision processes: insights and challenges. In *Proceedings of the 8th conference on Theoretical aspects of rationality and knowledge.*

Boye, J. (2007). Dialogue Management for Automatic Troubleshooting and Other Problem-solving Applications. In *Proceedings SIGDial'07.*

Branavan, S. R. K., Chen, H., Zettlemoyer, L. S., and Barzilay, R. (2009). Reinforcement learning for mapping instructions to actions. In *Proc. ACL 2009.*

Branavan, S. R. K., Chen, H., Zettlemoyer, L. S., and Barzilay, R. (2010). Reading between the lines: Learning to Map High-level Instructions to Commands. In *Proc. ACL 2010, Uppsala, Sweden.*

Branigan, H. P., Pickering, M. J., Pearson, J., and McLean, J. F. (2010). Linguistic alignment between people and computers. *Journal of Pragmatics*, 42(9):2355–2368.

Brennan, S. E. (1991). Conversation with and through computers. *User Modeling and User-Adaptive Interaction*, 1(1):67–86.

Brusilovsky, P. and Maybury, M. (2002). From adaptive hypermedia to the adaptive web. *Communications of the ACM*, 45(5):30–33.

Brusk, J., Lager, T., Hjalmarsson, A., and Wik, P. (2007). DEAL: dialogue management in SCXML for believable game characters. In *Proceedings of the 2007 conference on Future Play*.

Buschmeier, H., Bergmann, K., and Kopp, S. (2010). Modelling and Evaluation of Lexical and Syntactic Alignment with a Priming-Based Microplanner. In Krahmer, E. and Theune, M., editors, *Empirical Methods in Natural Language Generation*, volume 5980 of *LNCS*. Springer, Berlin / Heidelberg.

Busemann, S. and Horacek, H. (1998). A flexible shallow approach to text generation. In *Proc. 9th International Workshop on Natural Language Generation, Canada*.

Butenkov, D. (2009). Towards a Flexible User Simulation for Evaluating Spoken Dialogue Systems. In *Human-Computer Interaction - INTERACT 2009*, volume 5727/2009 of *LNCS*, pages 880–883. Springer, Berlin / Heidelberg.

Callaway, C. B., Dzikovska, M. O., Farrow, E., Marques-Pita, M., Matheson, C., and Moore, J. D. (2007). The Beetle and BeeDiff Tutoring Systems. In *Proceedings of the 2007 Workshop on Spoken Language Technology for Education (SLaTE), Farmington, Pennsylvania, USA, September 2007.*

Carberry, S. (1983). Tracking User Goals in an Information-Seeking Environment. In *Proceedings of AAAI.*

Carberry, S. (1990). *Plan Recognition in Natural Language Dialogue*. MIT Press, Cambridge, MA, USA.

Carberry, S., Chu-Carroll, J., and Elzer, S. (1999). Constructing and Utilizing a Model of User Preferences in Collaborative Consultation Dialogues. *Computational Intelligence Journal*, 15(3):185–217.

Carenini, G. and Moore, J. D. (2001). An Empirical Study of the Influence of User Tailoring on Evaluative Argument Effectiveness. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001), Seattle, USA*.

Cawsey, A. (1993). User Modelling in Interactive Explanations. *User Modeling and User-Adapted Interaction*, 3(3):221–247.

Cheng, H., Bratt, H., Mishra, R., Shriberg, E., Upson, S., Chen, J., Weng, F., Peters, S., Cavedon, L., and Niekrasz, J. (2004). A Wizard of Oz Framework for Collecting Spoken Human-Computer Dialogs. In *Proc. Intl. Conf. on Spoken Language Processing, Jeju, Korea*.

Cheng, H., Poesio, M., Henschel, R., and C., M. (2001). Corpus-based np modifier generation. In *Proceedings of NAACL 01, Morristown, NJ, USA*.

Chin, D. (1989). KNOME: Modeling what the user knows in UC. In Kobsa, A. and Wahlster, W., editors, *User Models in Dialog Systems*. Springer-Verlag, London.

Chin, D. N. (1986). User Modelling in UC, the UNIX Consultant. In *Proceedings of CHI'86, Boston*.

Chu-Carroll, J. (2000). Mimic: An adaptive mixed initiative spoken dialogue system for information queries. In *Proceedings of the 6th Conference on Applied Natural Language Processing (ANLP)*.

Chu-Carroll, J. and Nickerson, J. (2000a). Evaluating automatic dialogue strategy adaptation for a spoken dialogue system. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Chu-Carroll, J. and Nickerson, J. S. (2000b). Evaluating Automatic Dialogue Strategy Adaptation for a Spoken Dialogue System. In *Proceedings of NAACL 2000*.

Chung, G. (2004). Developing a flexible spoken dialog system using simulation. In *Proceedings of ACL 2004, Spain*.

Church, K. and Gale, W. (1991). A Comparison of the Enhanced Good-Turing and Deleted Estimation Methods for Estimating Probabilities of English Bigrams. *Computer Speech and Language*, 5(1).

Clark, H. H. (1996). *Using Language*. Cambridge University Press, Cambridge.

Clark, H. H. and Murphy, G. L. (1982). Audience design in meaning and reference. In Leny, J. F. and Kintsch, W., editors, *Language and comprehension*. North-Holland Publishing Company, Amsterdam: North-Holland.

Cohen, P. R. (1981). The need for referent identification as a planned action. In *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 1*, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Cohen, P. R. and Perrault, C. R. (1979). Elements of a plan-based theory of speech acts. *Cognitive Science*, 3(3):177212.

Cohen, R. and Jones, M. (1989). Incorporating User Models into Expert Systems for Educational Diagnosis. In Kobsa, A. and Wahlster, W., editors, *User Models in Dialog Systems*, pages 313–333. Springer, Berlin, Heidelberg.

Cohen, W. (1995). Fast effective rule induction. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann.

Cooke, M. W., Wilson, S., Cox, P., and Roalfe, A. (2000). Public understanding of medical terminology: non-English speakers may not receive optimal care. *Emergency Medicine Journal*, 17(2):119–121.

Cuayahuitl, H. (2009). *Hierarchical Reinforcement Learning for Spoken Dialogue Systems*. Ph.D. thesis, University of Edinburgh, UK.

Cuayahuitl, H., Renals, S., Lemon, O., and Shimodaira, H. (2005). Human-Computer Dialogue Simulation Using Hidden Markov Models. In *Proceedings of ASRU 2005*.

Dale, R. (1988). *Generating Referring Expressions in a Domain of Objects and Processes*. Ph.D. thesis, University of Edinburgh.

Dale, R. (1989a). Cooking up referring expressions. In *Proceedings ACL-1989*.

Dale, R. (1989b). Generating recipes: An overview of EPICURE. In *Proceedings of 2nd European Workshop on Natural Language Generation, Edinburgh, UK*.

Dale, R. and Haddock, N. (1991). Content Determination in the Generation of Referring Expressions. *Computational Intelligence*, 7(4):252–265.

Das, S. A., Datar, M., Garg, A., and Rajaram, S. (2007). Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*.

Demberg, V. and Moore, J. D. (2006). Information Presentation in Spoken Dialogue Systems. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Dethlefs, N. and Cuayahuitl, H. (2010). Hierarchical Reinforcement Learning for Adaptive Text Generation. In *Proceedings of the 6th International Natural Language Generation Conference*.

Digman, J. M. (1990). Personality structure: Emergence of the five-factor model. *Annual Review of Psychology*, 41.

Duboue, P. A. and McKeown, K. R. (2003). Statistical acquisition of content selection rules for natural language generation. In *Proceedings of EMNLP'03*.

Dzikovska, M. O., Callaway, C., Farrow, E., Marques-Pita, M., Matheson, C., and Moore, J. D. (2007). Adaptive tutorial dialogue systems using deep nlp techniques. In *NAACL '07: Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations on XX*, pages 5–6, Morristown, NJ, USA. Association for Computational Linguistics.

Eberts, R. (1994). *User Interface Design*. Prentice Hall.

Eckert, W., Levin, E., and Pieraccini, R. (1997). User Modeling for Spoken Dialogue System Evaluation. In *Proceedings of ASRU97*.

Filisko, E. and Seneff, S. (2006). Learning Decision Models in Spoken Dialogue Systems via User Simulation. In *Proceedings of the AAAI workshop on Statistical and Empirical Approaches for Spoken Dialogue Systems*.

Forbes-Riley, K. and Litman, D. (2009a). Adapting to student uncertainty improves tutoring dialogues. In *Proceedings 14th International Conference on Artificial Intelligence in Education (AIED), Brighton, UK*.

Forbes-Riley, K. and Litman, D. (2009b). A user modeling-based performance analysis of a wizarded uncertainty-adaptive dialogue system corpus. In *Proceedings Interspeech, Brighton, UK*.

Forbes-Riley, K. and Litman, D. (2010). Designing and evaluating a wizarded uncertainty-adaptive spoken dialogue tutoring system. *Computer Speech and Language*.

Frampton, M. (2008). *Context Models for Dialogue Strategy Learning*. Ph.D thesis, University of Edinburgh.

Fraser, N. and Gilbert, G. N. (1991). Simulating speech systems. *Computer Speech and Language*, 5:81–99.

Garoufi, K. and Koller, A. (2010). Automated planning for situated natural language generation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL), Uppsala, Sweden*.

Gatt, A. and Belz, A. (2008). Attribute Selection for Referring Expression Generation: New Algorithms and Evaluation Methods. In *Proceedings INLG-2008*.

Gatt, A. and van Deemter, K. (2009). Generating plural NPs in discourse: Evidence from the GNOME corpus. In *Proceedings of the Workshop on Production of Referring Expressions: Bridging Computational and Psycholinguistic Approaches (PRE-CogSci-09)*.

Georgila, K., Henderson, J., and Lemon, O. (2005). Learning User Simulations for Information State Update Dialogue Systems. In *Proceedings of Eurospeech/Interspeech*.

Georgila, K., Henderson, J., and Lemon, O. (2006). User Simulation for Spoken Dialogue System: Learning and Evaluation. In *Proceedings of ICSLP 2006*.

Goldberg, E., Driedgar, N., and Kittredge, R. (1994). Using Natural Language Processing to produce Weather Forecasts. *IEEE Expert*, 9:45–53.

Golland, D., Liang, P., and Klein, D. (2010). A Game-Theoretic Approach to Generating Spatial Descriptions. In *Proceedings of EMNLP 2010*.

Graesser, A. C., Wiemer-Hastings, P., WiemerHastings, P., and Kreuz, R. (1999). AutoTutor: A simulation of a human tutor. *Cognitive Systems Research*, 1:35–51.

Greenbacker, C. F. and McCoy, K. F. (2009). Feature Selection for Reference Generation as Informed by Psycholinguistic Research. In *Proceedings of the Workshop on Production of Referring Expressions: Bridging Computational and Psycholinguistic Approaches (PRE-CogSci-09).*

Grice, H. P. (1975). Logic and Conversation. *Syntax and Semantics: Vol 3, Speech Acts*, pages 43–58.

Griol, D., Callejas, Z., and López-Cózar, R. (2010). Statistical dialog management methodologies for real applications. In *Proceedings of the SIGDIAL 2010 Conference*, pages 269–272, Tokyo, Japan. Association for Computational Linguistics.

Guinn, C. I. (1998). An Analysis of Initiative Selection in Collaborative Task-Oriented Discourse. *User Modelling and User-Adapted Interaction*, 8(3-4):255–314.

Hajdinjak, M. and Miheli, F. (2003). The Wizard of Oz System for Weather Information Retrieval. In *Proceedings of the 6th International Conference TSD*, pages 400–405.

Hassel, L. and Hagen, E. (2005). Adaptation of an Automotive Dialogue System to User's Expertise. In *Proceedings of SIGDial 2005, Portugal.*

Hayes, P. J. and Rosner, M. (1976). Ully: A Program for Handling Conversations. In *Proceedings of AISB (ECAI).*

Heeman, P. and Hirst, G. (1995). Collaborating on referring expressions. *Computational Linguistics*, 21(3):351382.

Heller, D., Skovbroten, K., and Tanenhaus, M. K. (2009). Experimental evidence for speakers sensitivity to common vs. privileged ground in the production of names. In *Proceedings PRE-CogSci 2009.*

Henderson, J., Lemon, O., and Georgila, K. (2005). Hybrid Reinforcement/Supervised Learning for Dialogue Policies from COMMUNICATOR data. In *IJCAI workshop on Knowledge and Reasoning in Practical Dialogue Systems'05.*

Henderson, J., Lemon, O., and Georgila, K. (2008). Hybrid reinforcement / supervised learning of dialogue policies from fixed datasets. *Computational Linguistics*, 34(4).

Hernandez, F., Gaudioso, E., and Boticario, J. G. (2003). A Multiagent Approach to Obtain Open and Flexible User Models in Adaptive Learning Communities. In *User Modeling 2003*, volume 2702/2003 of *LNCS*. Springer, Berlin / Heidelberg.

Hernandez, F., Gaudioso, E., and Boticario, J. G. (2004). A Reinforcement Learning Approach to Achieve Unobtrusive and Interactive Recommendation Systems for Web-Based Communities. In *Adaptive Hypermedia and Adaptive Web-Based Systems*, volume 3137/2004 of *LNCS*, pages 409–412. Springer, Berlin / Heidelberg.

Hinds, P. (1999). The Curse of Expertise: The effects of expertise and debiasing methods on predictions of novice performance. *Experimental Psychology: Applied*, 5(2):205–221.

Horvitz, E., Breese, J., Heckerman, D., Hovel, D., and Rommelse, K. (1998). The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, San Francisco*.

Horvitz, E., Jacobs, A., and Hovel, D. (1999). Attention-sensitive Alerting. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm*.

Iordanskaja, L., Kim, M., Kittredge, R., Lavoie, B., and Polguere, A. (1992). Generation of extended bilingual statistical reports. In *Proceedings of the 14th conference on Computational linguistics*, pages 1019–1023, Morristown, NJ, USA. Association for Computational Linguistics.

Isard, A., Brockmann, C., and Oberlander, J. (2006). Individuality and alignment in generated dialogues. In *Proceedings of the Fourth International Natural Language Generation Conference*, Proceedings of INLG '06.

Isard, A., Oberlander, J., Androutsopoulos, I., and Matheson, C. (2003). Speaking the users' languages. *IEEE Intelligent Systems*, 18:40–45.

Ishizaki, M., Crocker, M., and Mellish, C. (1999). Exploring Mixed-Initiative Dialogue using Computer Dialogue Simulation. *User Modelling and User-Adapted Interaction*, 9(1-2):79–91.

Issacs, E. A. and Clark, H. H. (1987). References in conversations between experts and novices. *Journal of Experimental Psychology: General*, 116:26–37.

Jameson, A. (1995). Numerical Uncertainty Management in User and Student Modeling: An Overview of Systems and Issues. *User Modeling and User-Adapted Interaction*, 5(3-4):193–251.

Jameson, A., Gromann-Hutter, B., March, L., and Rummer, R. (2000). Creating an Empirical Basis for Adaptation Decisions. In *Proceedings of the International Conference on Intelligent User Interfaces. New Orleans, Louisiana*.

Johnston, M., Bangalore, S., Vasireddy, G., Stent, A., Ehlen, P., Walker, M., Whittaker, S., and Maloor, P. (2002). MATCH: An architecture for multimodal dialogue systems. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Jokinen, K. (2006). Adaptation and user expertise modelling in athosmail. *Univers. Access Inf. Soc.*, 4(4):374–392.

Jokinen, K. and Kanto, K. (2004). User expertise modelling and adaptivity in a speech-based e-mail system. In *Proceedings of the ACL-04, Barcelona*.

Jokinen, K., Rissanen, J., Kernen, H., and Kanto, K. (2002). Learning interaction patterns for adaptive user interfaces. In *Proceedings of the 7th ERCIM Workshop User Interfaces for all 2002. Paris.*

Jordan, P., Makatchev, M., Pappuswamy, U., VanLehn, K., and Albacete, P. (2006). A Natural Language Tutorial Dialogue System for Physics. In *Proceedings of FLAIRS'06, 2006.*

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237285.

Kalyuga, S. (2003). The Expertise Reversal Effect. *Educational Psychologist*, 38:23–31.

Kalyuga, S. (2007). Expertise Reversal Effect and Its Implications for Learner-Tailored Instruction. *Journal of Educational Psychology Review*, 19(4):509–539.

Kalyuga, S. (2009). Instructional designs for the development of transferable knowledge and skills: A cognitive load perspective. *Comput. Hum. Behav.*, 25(2):332–338.

Kass, R. (1991). Building a User Model Implicitly from a Cooperative Advisory Dialogue. *User Modeling and User-Adapted Interaction*, 1:203–258.

Kavcic, A. (2000). The Role of User Models in Adaptive Hypermedia Systems. In *Proceedings of the 10th Mediterranean Electrotechnical Conference MEleCon 2000, May, Lemesos, Cyprus.*

Kay, J., Kummerfeld, B., and Lauder, P. (2002). Personis: A server for user models. In *Proceedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'2002)*, pages 201–212.

Keizer, S., Gasic, M., Jurcicek, F., Mairesse, F., Thomson, B., Yu, K., and Young, S. (2010). Parameter estimation for agenda-based user simulation. In *Proceedings of SIGDial 2010, Tokyo.*

Kersting, K. and De Raedt, L. (2003). Logical Markov decision programs. In *Proceedings of the IJCAI'03 Workshop on Learning Statistical Models of Relational Data.*

Kittredge, R., Korelsky, T., and Rambow, O. (1991). On the need for domain communication knowledge. *Computational Intelligence*, 7(4):305–314.

Kobsa, A. and Wahlster, W. (1989). *User Models in Dialog Systems.* Berlin: Springer Verlag.

Koch-Weser, S., Dejong, W., and Rudd, R. E. (2009). Medical word use in clinical encounters. *Health Expectations*, 12(4):371–382.

Koller, A. and Stone, M. (2007). Sentence generation as planning. In *Proceedings of ACL-07, Prague*.

Komatani, K. and Okuno, H. G. (2010). Online Error Detection of Barge-In Utterances by Using Individual Users Utterance Histories in Spoken Dialogue System. In *Proceedings of SIGDial 2010*.

Komatani, K., Ueno, S., Kawahara, T., and Okuno, H. G. (2003). Flexible Guidance Generation using User Model in Spoken Dialogue Systems. In *Proceedings ACL'03*.

Komatani, K., Ueno, S., Kawahara, T., and Okuno, H. G. (2005). User Modeling in Spoken Dialogue Systems to Generate Flexible Guidance. *User Modeling and User-Adapted Interaction*, 15-1:169–183.

Krahmer, E. and Theune, M. (2002). Efficient context-sensitive generation of referring expressions. In van Deemter, K. and Kibble, R., editor, *Information Sharing: Reference and Presupposition in Language Generation and Interpretation*, pages 223–264. CSLI, Stanford, CA.

Krahmer, E., van Erk, S., and Verleg, A. (2003). Graph-based Generation of Referring Expressions. *Computational Linguistics*, 29(1):53–72.

Kullback, S. (1959). *Information theory and statistics.* John Wiley and Sons, NY.

Kullback, S. (1987). Letter to the Editor: The KullbackLeibler distance. *The American Statistician*, 41 (4):340341.

Kullback, S. and Leibler, R. A. (1951). On Information and Sufficiency. *Annals of Mathematical Statistics*, 22 (1):7986.

Langkilde, I. and Knight (1998). Generation that Exploits Corpus-Based Statistical Knowledge. In *Proceedings of 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*.

Lemon, O. (2008). Adaptive Natural Language Generation in Dialogue using Reinforcement Learning. In *Proceedings of SEMdial'08*.

Lemon, O., Bracy, A., Gruenstein, A., and Peters, S. (2001). The witas multimodal dialogue system i. In *Proceedings EuroSpeech 2001.*

Lemon, O., Georgila, K., and Stuttle, M. (2006). An ISU dialogue system exhibiting reinforcement learning of dialogue policies: generic slot-filling in the TALK in-car system. In *Proceedings of EACL (demo session), 2006*.

Lemon, O., Janarthanam, S., and Rieser, V. (2010). Generation under Uncertainty. In *Proceedings of INLG 2010*.

Lerner, E. B., Jehle, D. V., Janicke, D. M., and Moscati, R. M. (2000). Medical communication: do our patients understand? *Americal Journal of Emergency Medicine*, 18(7):764–6.

Levin, E., Pieraccini, R., and Eckert, W. (1997). Learning Dialogue Strategies within the Markov Decision Process Framework. In *Proceedings of ASRU97*.

Levin, E., R. Pieraccini, R., and Eckert, W. (2000). A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Trans. on Speech and Audio Processing*, 8(1):11–23.

Lin, B. S. and Lee, L. S. (2001). Computer aided analysis and design for spoken dialogue systems based on quantitative simulations. *IEEE Trans Speech Audio Process*, 9(5):534–548.

Litman, D. and Pan, S. (1999). Empirically evaluating an adaptable spoken dialogue system. In *Proceedings of the 7th International Conference on User Modeling*.

Litman, D. and Pan, S. (2002). Designing and evaluating an adaptive spoken dialogue system. *User Modeling and User-Adapted Interaction*, 12(2-3):111–137.

Litman, D. and Silliman, S. (2004). ITSPOKE: An Intelligent Tutoring Spoken Dialogue System. In *Proceedings of the Human Language Technology Conference: 4th Meeting of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL), Boston, MA*.

Littman, M. L. (1996). *Algorithms for Sequential Decision Making*. Ph.D. thesis, Brown University.

Lopez-Cozar, R., Callejas, Z., Gea, M., and Montoro, G. (2005). Multimodal, multilingual and adaptive dialogue systemfor ubiquitous interaction in educational space. In *Proceedings of Applied Spoken Language Interaction in DistributedEnvironments (ASIDE), Aalborg (Denmark)*.

Lopez-Cozar, R., de la Torre, A., Segura, J. C., and Rubio, A. J. (2003). Assessment of Dialogue Systems by Means of a New Simulation Technique. *Speech Communication*, 40(3):387–407.

Lucas, S. E. (2003). *The Art of Public Speaking*. McGraw-Hill Humanities/Social Sciences/Languages; 8 edition.

MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*.

Magnini, B. and Strapparava, C. (2010). Improving User Modelling with Content-Based Techniques. In *User Modelling 2001*, volume 2109/2010 of *LNCS*, pages 74–83. Springer, Berlin / Heidelberg.

Mairesse, F. and Walker, M. (2007). PERSONAGE: Personality Generation for Dialogue. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL), Prague*.

Mairesse, F. and Walker, M. (2008). A Personality-based Framework for Utterance Generation in Dialogue Applications. In *Proceedings of the AAAI Spring Symposium on Emotion, Personality, and Social Behavior, Palo Alto*.

Mairesse, F. and Walker, M. (2010). Towards Personality-Based User Adaptation: Psychologically Informed Stylistic Language Generation. *User Modeling and User-Adapted Interaction*, 20:3.

Malmstrom, K., Munday, L., and Sitte, J. (1996). Reinforcement Learning of a Path-Finding Behaviour by a Mobile Robot. In *Australian New Zealand Conference on Intelligent Information Systems, Adelaide*.

Martins, A. C., Faria, L., Vaz de Carvalho, C., and Carrapatoso, E. (2008). User Modeling in Adaptive Hypermedia Educational Systems. *Educational Technology & Society*, 11 (1):194–207.

McCoy, K. (1985). *Correcting Object-related Misconceptions*. Ph.D thesis, MIT.

McCoy, K. F. (1989). Generating Context Sensitive Responses to Object-Related Misconceptions. *Artificial Intelligence*, 41:157–195.

McDonald, D. D. (1980). *Natural Language Production as a Process of Decision Making Under Constraint*. Ph.D thesis, MIT.

McKeown, K., Robin, J., and Tanenblatt, M. (1993). Tailoring Lexical Choice to the User's Vocabulary in Multimedia Explanation Generation. In *Proceedings ACL 1993*.

McKeown, K. R. (1985). *Text Generation*. Cambridge University Press.

McMurrey, D. C. (2001). *Power Tools for Technical Communication*. Wadsworth Publishing.

McRoy, S. W., Channarukul, S., and A., S. S. (2003). An augmented template-based approach to text realization. *Natural Language Engineering*, 9(4):381420.

McTear, M. (1993). User Modelling for Adaptive Computer Systems: A Survey of Recent Developments. *Artificial Intelligence Review. (Special issue on User Modelling, edited by M McTear)*, 7:157–184.

Molich, R. and Nielsen, J. (1990). Improving a Human-Computer Dialogue. *Communications of the ACM*, 33-3:338–348.

Moore, J. D., Foster, M. E., Lemon, O., and White, M. (2004). Generating Tailored, Comparative Descriptions in Spoken Dialogue. In *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Sociey Conference*.

Moukas, A. and Maes, P. (1998). Amalthaea: An evolving multiagent information filtering and discovery system for the www. *Autonomous Agents and Multi-Agent Systems*, 1.

Netto, S., Leite, V., Silva, A., Paiva, A., and Neto, A. (2008). Application on Reinforcement Learning for Diagnosis Based on Medical Image. In Weber, C., Elshaw, M., and Mayer, N. M., editors, *Reinforcement Learning*. I-Tech Education and Publishing.

Nguyen, L. and Do, P. (2008). Learning Model in Adaptive Learning. In *Proceedings of World Academy of Science, Engineering and Technology*.

Nguyen, L. and Do, P. (2009). Combination of Bayesian Network and Overlay Model in User Modeling. In *Proceedings of the 9th International Conference on Computational Science*, volume 5545/2009 of *LNCS*, pages 5–14. Springer, Berlin / Heidelberg.

Nielsen, J. (1993). *Usability Engineering*. Academic Press Professional, Boston.

Oberlander, J. and Brew, C. (2000). Stochastic Text Generation. *Philosophical Transactions of the Royal Society of London*, Series A, 358:1373–1385.

Oberlander, J. and Dale, R. (1991). Generating expressions referring to eventualities. In *Proceedings of the 13th Annual Conference of the Cognitive Science Society*, pages 67–72.

O'Donnell, M., Mellish, C., Oberlander, J., and Knott, A. (2001). ILEX: An architecture for a dynamic hypertext generation system. *Natural Language Engineering*, 7:225–250.

Oh, A. and Rudnicky, A. (2002). Stochastic natural language generation for spoken dialog systems. *Computer Speech and Language*, 16(3-4):387–407.

Paris, C. L. (1984). Determining the Level of Expertise. In *First annual workshop on Theoritical Issues in Conceptual Information Processing, Atlanta*.

Paris, C. L. (1987). *The Use of Explicit User Models in Text Generation: Tailoring to a User's Level of Expertise*. Ph.D. thesis. Columbia University, Department of Computer Science, New York.

Paris, C. L. (1988). Tailoring Object Descriptions to a User's Level of Expertise. *Computational Linguistics*, 14(3):64–78.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann Publishers.

Pearson, J., Hu, J., Branigan, H. P., Pickering, M. J., and Nass, C. (2006). Adaptive language behavior in HCI: how expectations and beliefs about a system affect users' word choice. In *Proceedings of the SIGCHI conference on Human Factors in computing systems, Montral*.

Perkowitz, M. and Etzioni, O. (2000). Towards adaptive web sites: Conceptual framework and case study. *ARTIFICIAL INTELLIGENCE*, 118:245–275.

Pickering, M. J. and Garrod, S. (2004). Toward a mechanistic psychology of dialogue. *Behavioral and Brain Sciences*, 27:169–225.

Pietquin, O. and Dutoit, T. (2003). Aided design of finite-state dialogue management systems. In *Proceedings of the 2003 International Conference on Multimedia and Expo - Volume 3 (ICME '03) - Volume 03*.

Pietquin, O. and Dutoit, T. (2006). A probabilistic framework for dialog simulation and optimal strategy learning. *IEEE Transactions on Audio, Speech and Language Processing*, 14(2):589599.

Poesio, M., Henschel, R., Hitzeman, J., and Kibble, R. (1999). Statistical NP Generation: A First Report. In *Proceedings of the ESSLLI Workshop on NP Generation. Utrecht.*

Polifroni, J., Chung, G., and Seneff, S. (2003). Towards the automatic generation of mixed-initiative dialogue systems from web-content. In *Proceedings of Eurospeech'03*.

Porzel, R., Scheffler, A., and Malaka, R. (2006). How entrainment increases dialogical efficiency. *Proceedings of Workshop on on Effective Multimodal Dialogue Interfaces, Sydney*.

Quarteroni, S. and Manandhar, S. (2006a). Adaptivity in Question Answering with User Modelling and a Dialogue Interface. In *Proceedings of EACL 2006*.

Quarteroni, S. and Manandhar, S. (2006b). User modelling for adaptive question answering and information retrieval. In Sutcliffe, G. and Goebel, R., editors, *Proceedings of FLAIRS*. AAAI Press.

Rambow, O. (1990). Domain Communication Knowledge. In *Proceedings of the Fifth International Workshop on Natural Language Generation 1990*.

Raskutti, B., Beitz, A., and Ward, B. (1997). A Feature-based Approach to Recommending Selections based on Past Preferences. *User Modeling and User-Adapted Interaction*, 7(3):179–218.

Raux, A., Langner, B., Black, A., and Eskenazi, M. (2003). LET'S GO: Improving Spoken Dialog Systems for the Elderly and Non-natives. In *Proceedings of Eurospeech 2003, Geneva, Switzerland.*

Raux, A., Langner, B., Black, A., and Eskenazi, M. (2005). Let's Go Public! Taking a Spoken Dialog System to the Real World. In *Proceedings of Interspeech 2005 (Eurospeech), Lisbon, Portugal.*

Reiter, E. (1991a). A New Model of Lexical Choice for Nouns. *Computational Intelligence*, 7:240–251.

Reiter, E. (1991b). Generating Descriptions that Exploit a User's Domain Knowledge. In Dale, R., Mellish, C., and Zock, M., editors, *Current Research in Natural Language Generation*, pages 257–285. Academic Press.

Reiter, E. and Dale, R. (1992). A Fast Algorithm for the Generation of Referring Expressions. In *Proceedings COLING-1992*.

Reiter, E. and Dale, R. (1995). Computational Interpretations of the Gricean Maxims in the Generation of Referring Expressions. *Cognitive Science*, 18:233–263.

Reiter, E. and Dale, R. (1997). Building Applied Natural-Language Generation Systems. *Journal of Natural-Language Engineering*, 3:57–87.

Reiter, E. and Dale, R. (2000). *Building Natural Language Generation Systems*. Cambridge University Press.

Reiter, E., Mellish, C., and Levine, J. (1995). Automatic Generation of Technical Documentation. *Applied Artificial Intelligence*, 9:259–287.

Renkl, A. and Atkinson, R. K. (2007). An example order for cognitive skill acquisition. In Ritter, F. E., Nerb, J. Lehtinen, E., and OShea, T. M., editors, *In order to learn: How the sequence of topics influences learning*, pages 95–105. New York: Oxford University Press.

Resnick, P., Iacovou, N., Sushak, M., Bergstrom, P., and Riedl, J. (1994). GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of the Conference on Computer Supported Collaborative Work*.

Rich, E. (1979). User Modeling via Stereotypes. *Cognitive Science*, 3:329–354.

Rich, E. (1989). Stereotypes and User Modeling. In *User models in dialog systems*.

Rich, E. (1999). Users are individuals: individualizing user models. *Int. J. Hum.-Comput. Stud.*, 51(2):323–338.

Rieser, V. (2008). *Bootstrapping Reinforcement Learning-based Dialogue Strategies from Wizard-of-Oz data*. Ph.D. thesis. University of Saarlandes, Germany.

Rieser, V. and Lemon, O. (2006). Cluster-based User Simulations for Learning Dialogue Strategies. In *Proceedings of Interspeech/ICSLP 2006*.

Rieser, V. and Lemon, O. (2009a). Learning human multimodal dialogue strategies. *Journal of Natural Language Engineering*.

Rieser, V. and Lemon, O. (2009b). Natural Language Generation as Planning Under Uncertainty for Spoken Dialogue Systems. In *Proceedings EACL 2009*.

Rieser, V. and Lemon, O. (2011). Learning and Evaluation of Dialogue Strategies for new Applications: Empirical Methods for Optimization from Small Data Sets. *Computational Linguistics*, 37:1.

Rogers, S., Fiechter, C., and Thompson, C. (2000). Adaptive User Interfaces for Automotive Environments. In *IEEE Intelligent Vehicles Symposium, Dearborn, MI*.

Rudnicky, A., Bennett, C., Black, A., Chotomongcol, A., Lenzo., K., Oh, A., and Singh, S. (2000). Tasks and domain specific modelling in the carnegie mellon communicator system. In *Proceedings of ICSLP. Vol. 2.*

Rummery, G. A. and Niranjan, M. (1994). On-line Q-learning Using Connectionist Systems. Technical Report CUED/F-INFENG/TR 166, Engineering Department, Cambridge University.

Salton, G. and McGill, M. (1983). *An Introduction to Modern Information Retrieval.* McGraw Hill.

Schatzmann, J., Georgila, K., and Young, S. J. (2005). Quantitative Evaluation of User Simulation Techniques for Spoken Dialogue Systems. In *Proceedings SIGdial workshop on Discourse and Dialogue '05.*

Schatzmann, J., Thomson, B., K., W., Ye, H., and Young, S. (2007a). Agenda-based User Simulation for Bootstrapping a POMDP Dialogue System. In *Proceedings of HLT/NAACL, Rochester.*

Schatzmann, J., Thomson, B., Weilhammer, K., Ye, H., and Young, S. J. (2007b). Agenda-based User Simulation for Bootstrapping a POMDP Dialogue System. In *Proceedingsof HLT/NAACL 2007.*

Schatzmann, J., Thomson, B., Ye, H., and Young, S. (2007c). Statistical User Simulation with a Hidden Agenda. In *Proceedings of 8th SIGDial Workshop on Discourse and Dialogue, Antwerp.*

Schatzmann, J., Weilhammer, K., Stuttle, M. N., and Young, S. J. (2006). A Survey of Statistical User Simulation Techniques for Reinforcement Learning of Dialogue Management Strategies. *Knowledge Engineering Review*, pages 97–126.

Scheffler, K. and Young, S. (2001). Corpus-based dialogue simulation for automatic strategy learning and evaluation. In *Proceedings NAACL Workshop on Adaptation in Dialogue Systems'01.*

Schlangen, D. (2004). Causes and strategies for requesting clarification in dialogue. In *Proceedings of the 5th SIGdial Workshop on Discourse and Dialogue (SIGDIAL 04), Boston.*

Schmidt, C. F., Sridharan, N. S., and Goodson, J. L. (1978). The Plan Recognition problem: An Intersection of Psychology and Artificial Intelligence. *Artificial Intelligence*, 11(1-2):45–83.

Seneff, S. and Polifroni, J. (2000). Dialogue management in the Mercury flight reservation system. In *Proceedings ANLP-NAACL Sat. Workshop, Seattle, 2000.*

Shapiro, D. and Langley, P. (2001). Using background knowledge to speed reinforcement learning. In *Proceedings of the Fifth International Conference on Autonomous Agents.*

Shapiro, D. and Langley, P. (2002). Separating skills from preference: Using learning to program by reward. In *Proceedings ICML-02*.

Siddharthan, A. and Copestake, A. (2004). Generating referring expressions in open domains. In *Proceedings ACL-04*.

Singh, S., Kearns, M., Litman, D., and Walker, M. (1999). Reinforcement learning for spoken dialogue systems. In *Proceedings of NIPS 1999*.

Singh, S., Litman, D., Kearns, M., and Walker, M. (2002). Optimizing Dialogue Management with Reinforcement Learning: Experiments with the NJFun System. *Journal of Artificial Intelligence Research*, 16:105–133.

Smart, W. and Kaelbling, L. (2002). Effective Reinforcement Learning for Mobile Robots. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automatons, Washington, DC*.

Smith, R. W. (1998). An Evaluation of Strategies for Selectively Verifying Utterance Meanings in Spoken Natural Language Dialogue. *International Journal of Human-Computer Studies*, 48:627–647.

Spiro, D. and Heidrich, F. (1983). Lay understanding of medical terminology. *Journal of Family Practice*, 17(2):277–9.

Stenchikova, S. and Stent, A. (2007). Measuring adaptation between dialogs. In *Proceedings of the 8th SIGdial Workshop on Discourse and Dialogue*.

Stoia, L., Shockley, D. M., Byron, D. K., and Fosler-Lussier, E. (2006). Noun phrase generation for situated dialogs. In *Proceedings of the Fourth International Natural Language Generation Conference*, pages 81–88, Sydney, Australia. Association for Computational Linguistics.

Strauss, P. M., Hoffmann, H., and Scherer, S. (2007). Evaluation and user acceptance of a dialogue system using Wizard-of-Oz recordings. In *Proceedings of 3rd IET International Conference on Intelligent Environments*.

Su, M., Huang, D., Chou, C., and Hsieh, C. (2004). A Reinforcement Learning Approach to Robot Navigation. In *Proceedings of the 2004 IEEE International Conference on Networking, Sensing and Control, Taipei, Taiwan*.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning*. MIT Press.

Taghipour, N., Kardan, A., and Ghidary, S. (2007). Usage-based web recommendations: a reinforcement learning approach. In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pages 113–120, New York, NY, USA. ACM.

Tetreault, J. and Litman, D. (2006). Using Reinforcement Learning to Build a Better Model of Dialogue State. In *Proceedings 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL), Trento, Italy*.

Theune, M., Klabbers, E., Odijk, J., de Pijper, J. R., and Krahmer, E. (2001). From data to speech: a general approach. *Natural Language Engineering*, 7(1):47–86.

Thompson, C. L. and Pledger, L. M. (1993). Doctor-Patient Communication: Is Patient Knowledge of Medical Terminology Improving? *Health Communication*, 5(2):89–97.

van Deemter, K. (2002). Generating referring expressions: Boolean extensions of the Incremental Algorithm. *Computational Linguistics*, 28(1):37–52.

van Deemter, K. (2009a). Utility and Language Generation: The case of Vagueness. *Journal of Philosophical Logic*, pages 607–632.

van Deemter, K. (2009b). What Game Theory can do for NLG: the case of vague language. In *Proceedings ENLG'09*.

van Deemter, K. and Odijk, J. (1997). Context modelling and the generation of spoken discourse. *Speech Communication*, 21(1/2):101121.

van Otterlo, M. (2004). Reinforcement Learning for Relational MDPs. In *Proceedings of Machine Learning Conference of Belgium and the Netherlands, BeNeLearn '04*.

Vogel, A. and Jurafsky, D. (2010). Learning to follow navigational directions. In *Proc. ACL 2010, Uppsala, Sweden*.

Wahlster, W. and Kobsa, A. (1989). User models in dialog systems. In *User Models in Dialog Systems*, pages 4–34. Springer-Verlag.

Walker, M., Kamm, C., and Litman, D. (2000). Towards developing general models of usability with paradise. *Natural Language Engineering*, 6(3-4):363–377.

Walker, M., Stent, A., Mairesse, F., and Prasad, R. (2007). Individual and domain adaptation in sentence planning for dialogue. *Journal of Artificial Intelligence Research*, 30:413–456.

Walker, M., Whittaker, S., Stent, A., Maloor, P., Moore, J. D., Johnston, M., and Vasireddy, G. (2002). Speech plans: generating evaluative responses in spoken dialogue. In *Proceedings of the International Conference on Natural Language Generation, New York*.

Walker, M., Whittaker, S., Stent, A., Maloor, P., Moore, J. D., Johnston, M., and Vasireddy, G. (2004). Generation and Evaluation of User Tailored Responses in Multimodal Dialogue. *Cognitive Science*, 28:811–840.

Watkins, C. (1989). *Learning from Delayed Rewards*. PhD Thesis, University of Cambridge, Cambridge, UK.

White, M. and Caldwell, T. (1998). EXEMPLARS: A practical, extensible framework for dynamic text generation. In *Proceedings of 9th International Workshop on Natural Language Generation, Canada.*, page 266275.

Whittaker, S., Walker, M., and Moore, J. (2002). Fish or Fowl: A Wizard of Oz Evaluation of Dialogue Strategies in the Restaurant Domain. In *Language Resources and Evaluation Conference*.

Wilensky, R., Arens, Y., and Chin, D. (1984). Talking to UNIX in English: an overview of UC. *ACM Communications*, 27:574–593.

Wilkinson, R. (2006). User Modeling for Information Retrieval on the Web. In *Second Workshop on Adaptive Systems and User Modeling on the World Wide Web*, pages 117–119.

William, R. S. (1983). Xplain: A system for creating and explaining expert consulting programs. *Journal of Artificial Intelligence*, 21(3):285–325.

Williams, J. (2007). Applying POMDPs to Dialog Systems in the Troubleshooting Domain. In *ProceedingsHLT/NAACL Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technology*.

Williams, J., Poupart, P., and Young, S. (2006). Partially Observable Markov decision processes with continuous observations for dialogue management. In *Proceedings SigDial'06*.

Williams, J. and Young, S. (2007). Partially Observable Markov Decision Processes for Spoken Dialog Systems. *Computer Speech and Language*, 21(20):231–422.

Williams, J. and Young, S. J. (2003). Using Wizard-of-oz Simulations to bootstrap Reinforcement Learning based Dialogue Management Systems. In *Proceedings SIGdial'03*.

Winterboer, A. and Moore, J. D. (2007). Evaluating Information Presentation Strategies for Spoken Recommendations. In *Proceedings of the ACM Conference on Recommender Systems*.

Witten, I. H. and Timothy, C. B. (1991). The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4).

Wittwer, J. and Renkl, A. (2008). Why instructional explanations often do not work: A framework for understanding the effectiveness of instructional explanations. *Educational Psychologist*, 43:49–64.

Wong, S. and Yao, Y. (1990). Query Formulation in Linear Retrieval Models. *Journal of the American Society for Information Science*, 41(5):334341.

Young, S. (2006). Using POMDPs for dialog management. In *Proceedings SLT'06*.

Zue, V., Seneff, S., Glass, J., Polifroni, J., Pao, C., Hazen, T., and Hetherington, L. (2000). JUPITER: A Telephone-Based Conversational Interface for Weather Information. *IEEE Transactions on Speech and Audio Processing*, 8-1.

Zukerman, I., Albrecht, D. W., and Nicholson, A. E. (1999). Predicting Users' Requests on the WWW. In *Proceedings of the Seventh International Conference on User Modeling, Banff, Canada*.