Adaptive Generation in Dialogue Systems Using Dynamic User Modeling

Srinivasan Janarthanam*
Heriot-Watt University

Oliver Lemon**
Heriot-Watt University

We address the problem of dynamically modeling and adapting to unknown users in resource-scarce domains in the context of interactive spoken dialogue systems. As an example, we show how a system can learn to choose referring expressions to refer to domain entities for users with different levels of domain expertise, and whose domain knowledge is initially unknown to the system. We approach this problem using a three-step process: collecting data using a Wizard-of-Oz method, building simulated users, and learning to model and adapt to users using Reinforcement Learning techniques.

We show that by using only a small corpus of non-adaptive dialogues and user knowledge profiles it is possible to learn an adaptive user modeling policy using a sense-predict-adapt approach. Our evaluation results show that the learned user modeling and adaptation strategies performed better in terms of adaptation than some simple hand-coded baseline policies, with both simulated and real users. With real users, the learned policy produced around a 20% increase in adaptation in comparison to an adaptive hand-coded baseline. We also show that adaptation to users' domain knowledge results in improving task success (99.47% for the learned policy vs. 84.7% for a hand-coded baseline) and reducing dialogue time of the conversation (11% relative difference). We also compared the learned policy with a variety of carefully hand-crafted adaptive policies that use the user knowledge profiles to adapt their choices of referring expressions throughout a conversation. We show that the learned policy generalizes better to unseen user profiles than these hand-coded policies, while having comparable performance on known user profiles.

We discuss the overall advantages of this method and how it can be extended to other levels of adaptation such as content selection and dialogue management, and to other domains where adapting to users' domain knowledge is useful, such as travel and healthcare.

* School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh.
E-mail: sc445@hw.ac.uk.
** School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh.
E-mail: o.lemon@hw.ac.uk.

Submission received: 16 November 2012; revised version received: 1 November 2013; accepted for publication: 18 January 2014.

doi:10.1162/COLI_a_00203

1. Introduction

A user-adaptive spoken dialogue system in a technical support domain should be able to generate instructions that are appropriate to the user's level of domain expertise (using appropriate referring expressions for domain entities, generating instructions with appropriate complexity, etc.). The domain knowledge of users is often unknown when a conversation starts. For instance, a caller calling a helpdesk to troubleshoot his laptop cannot be readily identified as a beginner, intermediate, or an expert in the domain. In natural human–human conversations, dialogue partners learn about each other and adapt their language to suit their domain expertise (Issacs and Clark 1987). This kind of adaptation is called "Alignment through Audience Design" (Clark and Murphy 1982; Bell 1984). Similar to this adaptive human behavior, a spoken dialogue system (SDS) must also be capable of observing the user's dialogue behavior, modeling his/her domain knowledge, and adapting accordingly.

Although there are several levels at which systems can adapt to users' domain knowledge, here we focus on adaptively choosing referring expressions that are used in technical instructions given to users. We also discuss how our model can later be extended to other levels of adaptation as well such as content selection and dialogue management. **Referring expressions** are linguistic expressions that are used to refer to domain objects of interest. Traditionally, the referring expression generation (REG) task includes selecting the type of expression (pronouns, proper nouns, common nouns, etc.), selecting attributes (color, type, size, etc.) and realizing them in the form of a linguistic expression (Reiter and Dale 2000). However, in this work, we focus on the user modeling aspects of referring expression generation. Our objective is to choose a referring expression (either a technical or a descriptive expression) that the user can understand easily and efficiently. For this, we build a dynamic user model to represent the user's domain knowledge that is estimated during the conversation. See Table 1 for some example utterances that we aim to generate using technical and descriptive expressions or a combination of the two types.

We present an approach to learning user-adaptive behavior by sensing partial information about the user's domain knowledge using unobtrusive information sensing moves, populating the user model, and then predicting the rest of the user's knowledge using reinforcement learning techniques. We present a three-step process to learning user-adaptive behavior in dialogue systems: data collection, building user simulations, and learning adaptive behavior using reinforcement learning. We show that the learned behavior performs better than a hand-coded adaptive behavior when evaluated with real users, by adapting to them and thereby enabling them to finish their task faster and more successfully. Our approach is corpus-driven and the system learns from a small corpus (only 12 dialogues) of non-adaptive human–machine interaction.

In Section 2, we analyze the problem of dynamic user modeling in spoken dialogue systems in detail. In Section 3, we present a technical support dialogue system that

Table 1
Variants of technical instructions to be generated by the system (with technical and descriptive expressions in *italics*).

1: Please plug one end of the *broadband cable* into the *broadband filter*.
2: Please plug one end of the *thin white cable with grey ends* into the *small white box*.
3: Please plug one end of the *broadband cable* into the *small white box*.

we use to build and experiment with our adaptive behavior learning model. We then discuss data collection, building user simulations, and learning adaptive behavior in Sections 4, 5, and 6. We present the results and analysis of the evaluations in Section 7. Finally, we present an experiment in simulation comparing the learned policy to a smart hand-coded policy, and discuss future work such as adapting at the level of content selection and dialogue management and adapting to dynamic knowledge profiles in Section 8.

2. Dynamic User Modeling

In order to adapt to the user, it is necessary for the system to have a model of the user's domain knowledge. This is currently taken into account by state-of-the-art REG algorithms by using an internal user model (UM). The UM determines whether the user would be able to relate the referring expression made by the system to the intended referent. To be more specific, it is used to estimate whether the user knows or would be able to determine whether an attribute-value pair applies to an object (Dale 1988; Reiter and Dale 1992, 1995; Krahmer and Theune 2002; Krahmer, van Erk, and Verleg 2003; Belz and Varges 2007; Gatt and Belz 2008; Gatt and van Deemter 2009). So, if the user model believes that the user cannot associate an attribute-value pair (e.g., $< category, recliner >$) to the target entity x, then it would return false. On the other hand, if he can instead associate the pair (e.g., $< category, chair >$) 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. However, these models are static and are predefined before run-time.

How can a system adapt when the user's knowledge is initially unknown at run-time? There are many cases when 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 user modeling strategies that can cope with unknown users. In order to deal with unknown users, a system should be able to do the following (Mairesse and Walker 2010):

- Sense: Learn about the user's domain knowledge during the course of interaction and populate the user model.
- Adapt: Adapt to the user by using the information in the user model.

A smarter system should be able to *predict* the user's domain knowledge from partial information sensed earlier. In our approach we aim to sense partial information, predict the rest, and adapt to the user. We refer to this approach as the **sense-predict-adapt** approach. 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 there are different underlying knowledge patterns for different types of users. Novice users may know technical expressions only for the most commonplace domain objects. Intermediate users may have knowledge of a few related concepts that form a subdomain within a larger domain (also called *local expertise* by Paris [1984]). Experts may know names for almost all the domain objects. Therefore, by knowing more about a user, the system can attempt to identify his/her expertise and more accurately predict the user's knowledge.

Sensing user knowledge can be done using explicit questions, or else implicitly by observing the user's responses to system instructions. In some dialogue systems,

explicit pre-task questions about the user's knowledge level in the task domain (e.g., broadband Internet connections, troubleshooting laptop issues) are used so that the system can produce adaptive utterances (McKeown, Robin, and Tanenblatt 1993). For instance, "Are you an expert or a novice?" 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 sensing questions at appropriate places in the conversation makes them less obtrusive. In large domains, a large number of explicit sensing questions would need to be asked, which could be unwieldy. In contrast, we aim to sense each user's domain knowledge implicitly by using expert technical (or "jargon") expressions within the interaction.

Another issue in user modeling 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 task domain experts (i.e., those with a good understanding of the task domain and its users) to hand-code the relationships 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 about when to seek information in order to partially populate the user model (Cawsey 1993). However, hand-coding such adaptation policies can be difficult for large and complex tasks that contain a large number of domain objects. 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). However, they require many annotated adaptive dialogues to train on. In gathering such a corpus, the expert should have exhibited adaptive behavior with users of all types. In addition, annotating a large number of dialogues to learn user modeling and adaptive strategies could be very expensive. Such an annotated corpus of expert–layperson interactions is a scarce resource.

Another issue is that domain experts suffer from what psychologists call *the curse of expertise* (Hinds 1999). This means that experts have difficulties communicating with non-experts because their own expertise distorts their predictions about non-experts. Such inaccurate predictions lead to underestimating or overestimating the non-expert's capabilities. Therefore, data collected using domain experts may not be ideal for systems to learn adaptation strategies from. Instead, it would be beneficial if such predictive rules for adaptation can be learned from non-adaptive dialogues, with little or no input from task domain experts. One reason for this is that non-adaptive dialogues may already be available or can be collected using existing troubleshooting scripts at technical call centers. Because data gathering using techniques like "Wizard of Oz" (WOZ) methods are expensive, we also investigate how adaptation strategies can be learned from limited data.

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

1. Unobtrusive dynamic user modeling by implicitly sensing and predicting user knowledge.
2. User modeling and adaptation using limited data and domain expertise.

Note that users may learn new referring expressions during the course of the interaction, and therefore the user's domain knowledge may be dynamically changing. However, we restrict ourselves to modeling and adapting to the initial knowledge state of the user. Modeling and adapting to a dynamically changing user knowledge state would be an interesting extension to our current work, and we discuss this later in the paper (see Section 8).

We chose to study the user modeling problem in a technical support dialogue system that chooses between two kinds of expressions: jargon and descriptive. **Jargon expressions** are very specific names given to an entity and are known only to experts in the domain (e.g., *broadband filter*). **Descriptive expressions**, as the name suggests, are more descriptive and identify the referent using attributes like shape, size and color, and so on (e.g., *small white box*). Although the choice between jargon and descriptive expressions may be motivated by many factors (learning gain, lexical alignment/entrainment, etc.), we focus on enabling users with different domain knowledge levels to identify the target entity efficiently. 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, Korelsky, and Rambow 1991). Therefore, this means that an expert user as defined in this article 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 she/he will be able to identify the domain entities using jargon expressions.

3. The Dialogue System

In order to explore the problem of dynamic user modeling, we built a "wizarded" technical support dialogue system that helps users to set up a home broadband connection. The dialogue system consists of a dialogue manager, a user modeling component, a natural language generation component, and a speech synthesizer. A human wizard recognizes user utterances and transcribes them into dialogue acts, which are sent to the dialogue manager. The dialogue manager decides the next dialogue move and sends a dialogue act to the natural language generation (NLG) module, which generates system utterances to be synthesized into speech by the speech synthesizer. The user modeling component takes input from the dialogue manager, dynamically models the user, and informs the NLG module which referring expressions to use based on its belief about the user's domain knowledge. The architecture of the system and its interaction with the user is shown in Figure 1.

3.1 Wizarded Speech Recognition and Language Understanding

We used a Wizard-of-Oz (WOZ) framework to both collect data and evaluate our learned model with real users. WOZ frameworks are often used to collect dialogues between real users and dialogue systems before actually implementing the dialogue system (Fraser and Gilbert 1991) . In this framework, participants interact with an expert human operator (known as a "wizard"), who is disguised as an automated dialogue system. These dialogue systems are called **wizarded dialogue systems** (Forbes-Riley and Litman 2010). WOZ systems have been used extensively to collect data to learn and test dialogue management policies (Whittaker, Walker, and Moore 2002; Hajdinjak and Miheli 2003; Cheng et al. 2004; Strauss, Hoffmann, and Scherer 2007; Rieser and Lemon 2011) and information presentation strategies (Demberg, Winterboer, and Moore 2011).

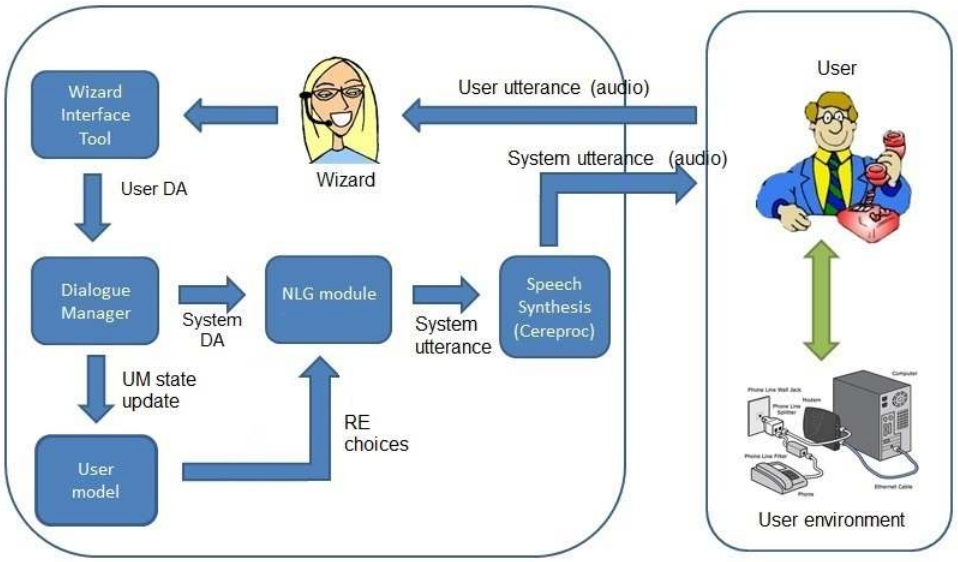

Figure 1
Wizarded spoken dialogue system.

In our system, the wizards played the role of intercepting, recognizing, and interpreting user speech into dialogue acts. Like Demberg, Winterboer, and Moore (2011), wizards in our set-up did not make dialogue management decisions. These were computed by the dialogue manager module based on the user dialogue act and the current dialogue state. Usually, in fully automated dialogue systems, automatic speech recognition (ASR) and natural language understanding (NLU) modules are used.

However, we use a human wizard to play the roles of ASR and NLU modules, so that we can focus on only the user modeling and NLG problem. ASR and NLU issues may make user modeling more complicated and their interaction should be studied carefully in future work.

The wizards were assisted by a tool called the Wizard Interpretation Tool (WIT), which was used by the wizard to interpret the user's utterances and generate the user dialogue acts (see Figure 2). The GUI was divided into several panels.

a. System Response Panel - This panel displayed the dialogue-system-generated response to the user's previous utterance and the system's referring expression (RE) choices for the domain objects in the utterance. This is done to serve as context for subsequent clarification requests from the user. It also displayed the strategy adopted by the system in the current dialogue and a visual indicator of whether the system response was being played back to the user.

b. Confirmation Request Panel - This panel enabled the wizard to handle issues in communication (e.g., noise). The wizard can ask the user to repeat, speak louder, confirm their responses, and so forth. Appropriate pre-recorded messages were played back to the user. There was also provision for the wizard to build custom messages and send them to the user. Custom messages were converted to speech and played back to the user.

c. Confirmation Panel - This panel enabled the wizard to handle confirmation questions from the user. The wizard can choose *yes* or *no* or build its own custom message. The message was converted to speech and played back to the user.

d. Annotation Panel - This panel enabled the wizard to annotate the content of the participant's utterances. Participant responses ranging from answers to questions, to acknowledging instructions, to requesting clarifications can be annotated. The annotated dialogue act is sent to the dialogue system for response. Table 2 shows the set of dialogue acts that can be annotated using this panel. In addition to these, other behaviors, like remaining silent or saying irrelevant things, were also accommodated. The WIT sent the generated dialogue act to the dialogue manager. For a more detailed description of the tool, please refer to Janarthanam and Lemon (2009).

3.2 Dialogue Manager

The dialogue manager identifies the next dialogue act ($A_{s,t}$ where t denotes turn number, s denotes system) to give to the user based on the dialogue management policy π_{dm}. The dialogue management policy is coded in the form of a finite state machine. It represents a series of instructions to be given to the user in order to set up a home broadband connection. In this dialogue task, the system provides instructions to either observe or manipulate the environment. The user's environment consists of several domain entities such as broadband and Ethernet cables, a broadband filter, sockets on the modem, and so forth. These are referred to by the NLG module using either jargon or descriptive expressions. If users ask for clarifications on jargon expressions, the system

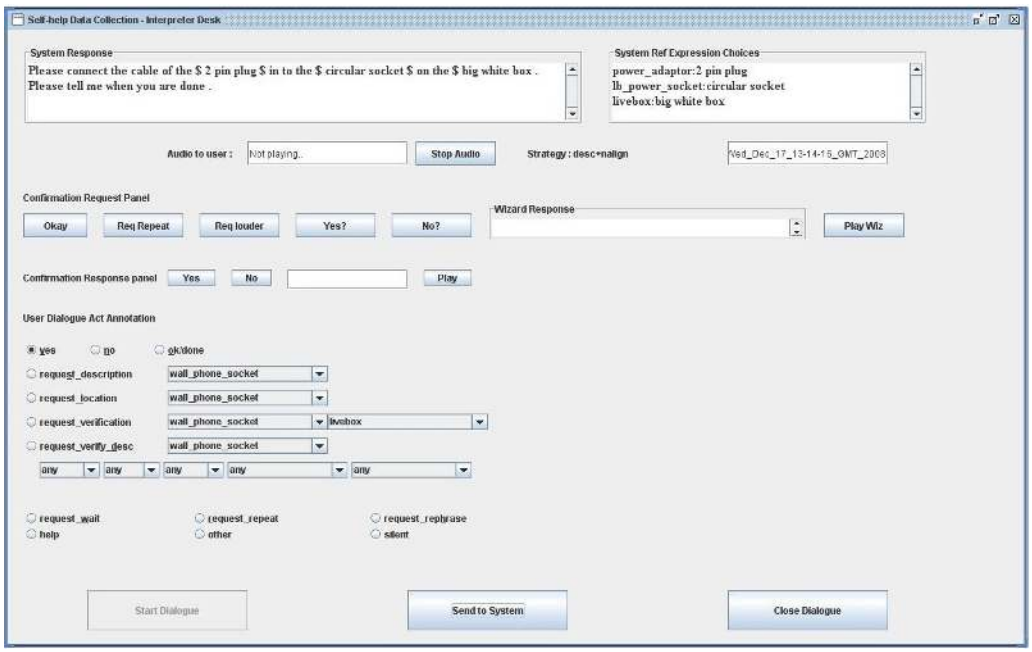

Figure 2
Wizard interpretation tool.

Table 2
User dialogue acts.

Dialogue Act	Example
yes	*Yes it is on*
no	*No, its not flashing*
ok	*Ok. I did that*
req_description	*Whats an Ethernet cable?*
req_location	*Where is the filter?*
req_verify_jargon	*Is it the Ethernet cable?*
req_verify_desc	*Is it the white cable?*
req_repeat	*Please repeat*
req_rephrase	*What do you mean?*
req_wait	*Give me a minute?*
help	*I need help*
other	*I had a bad morning*
silent	

clarifies (using the dialogue act *provide_clarification*) by giving information to enable the user to associate the expression with the intended referent. If users respond positively to the instructions given, the dialogue manager presents them with the next instruction, and so on. By "positive response," we mean that users answered observation questions correctly and they acknowledged following the manipulation instructions. For any other user response, the previous instruction is simply repeated. The dialogue manager is also responsible for updating and managing the system state $S_{s,t}$. The state $S_{s,t}$ is a set of variables that represents the current state of the conversation, which includes the state of the environment (i.e., how much of the broadband set-up has been finished).

3.3 User Modeling

A dynamic user modeling component incrementally updates a user model and informs other modules of the system about its estimates of the user (Kobsa and Wahlster 1989). In our system, the user modeling component maintains a user model $UM_{s,t}$, which represents the system's beliefs about what the user knows. The user model starts with a state where the system does not have any knowledge about the user. It is then updated dynamically based on the user's dialogue behavior during the conversation. Because the model is updated according to the user's behavior, it may be inaccurate if the user's behavior was itself uncertain. The user model is represented as a vector of n variables $(K_1, K_2 \ldots K_n)$. A user's knowledge of the technical name of each entity i is represented by variable K_i and takes one of the three values: *true*, *false*, and *unknown*.

The variables are updated using a simple user model update algorithm after the user's response to each turn. Initially each variable is set to *unknown*. If the user responds to an instruction containing the jargon expression for x with a clarification request, then K_x is set to *false* (assuming that the user did not know the technical

name for the entity x). If the user responds with an appropriate response to the system's instruction, K_x is set to *true*. Only the user's initial knowledge is recorded. This is based on the hypothesis (borne out by our evaluation) that an estimate of the user's initial knowledge helps to predict the user's knowledge of the rest of the entities.

In order to update the user model and inform the NLG module about its estimates of the user, the user modeling component recommends how an entity should be referred to in the system utterances. This behavior is generated by what is called the *UM policy* (π_{um}). This is the policy that we attempt to learn. We will later show how the UM policy interacts with other components of the dialogue system in order to populate the user model and estimate users' knowledge.

The *UM policy* (π_{um}) is defined as

$$\begin{aligned} &\pi_{um} : UM_{s,t} \rightarrow REC_{s,t} \\ &\quad \text{where } REC_{s,t} = \{(R_1, T_1), ..., (R_n, T_n)\} \end{aligned} \tag{1}$$

The referring expression choices $REC_{s,t}$ is a set of pairs identifying the referent R and the expression type T used in the current system utterance (s refers to system and t to turn number). For instance, the pair (*broadband_filter*, *desc*) represents the descriptive expression "small white box." Because the expression type is specified individually for each referent entity, it is possible to recommend jargon expressions for some entities and descriptive expressions for others in the same utterance.

The user modeling module can be operated in two modes. Given a UM policy (either hand-coded or learned), the task of this module is to recommend expressions specified in $REC_{s,t}$, depending on the user model state $UM_{s,t}$. We call this the **evaluation mode**. On the other hand, the user modeling module can operate as a learning agent in order to learn a UM policy, where it learns to associate the optimal RE choices to the UM states. We discuss the implementation of user modeling states in detail in Section 6.

3.4 NLG Module

The NLG module receives dialogue acts from the dialogue manager, retrieves an appropriate template, and picks appropriate referring expressions for each of the domain entities in the given dialogue act, based on recommendations from the user modeling component as described earlier. The NLG module then embeds the expressions into the templates to generate instructions.

3.5 Speech Synthesis Module

The utterances generated by the NLG module are then converted into speech by a speech synthesizer. We use the Cereproc Text-To-Speech[1] engine for this purpose.

1 `http://www.cereproc.com/`.

4. Data Collection

In this section, we present the first step of our three-step process for dynamic user modeling in interactive systems. Using the wizarded dialogue system presented previously, we collected a dialogue corpus from a number of users. Because we did not have an adaptive UM policy yet, we configured the user modeling module to generate two separate non-adaptive strategies: All-Jargon and All-Descriptive. In the All-Jargon policy, the system instructions only contained jargon expressions. Similarly, in the All-Descriptive policy, the system instructions contained only descriptive expressions. We collected half the dialogues with All-Jargon and the other half using All-Descriptive policies to analyze how users respond to jargon and descriptive expressions based on their domain knowledge.

Participants were presented with a box containing several objects (cables, modem, etc.), a phone, a phone socket, and a desktop computer, which were needed for a home broadband internet connection set-up. The modem consisted of several sockets that were used in this set-up (see Figure 3). The participants were asked to put these objects together in a specific pattern as instructed by the system. For example, the broadband cable must connect the modem to the phone socket, the Ethernet cable must be used to connect the desktop to the modem, and so on. The task had 16 steps to finish the broadband set-up and there were references to 13 domain entities, some of which were mentioned more than once in the dialogue. Users interacted with the system through a headset using speech.

We followed a six-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, the user's interaction with the physical environment, and the user satisfaction scores.

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, the internet, and dialogue systems.

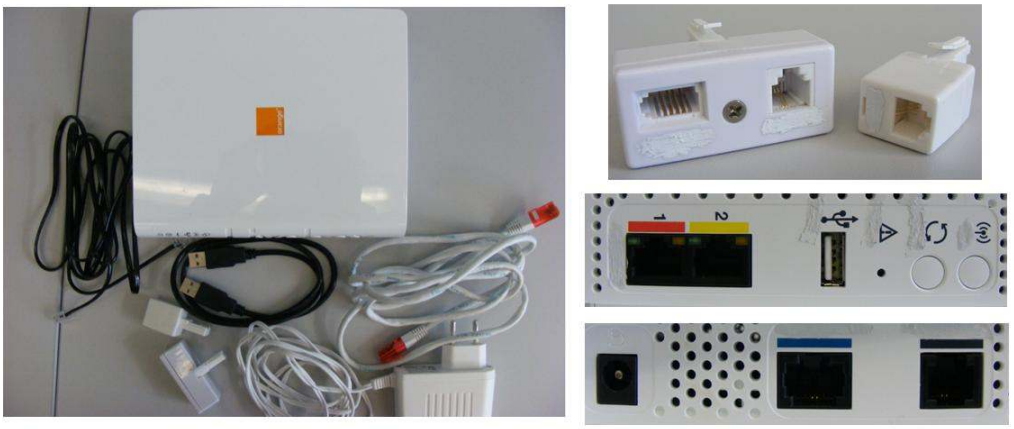

Figure 3
Domain objects for the broadband set-up.

Step 2. Knowledge pre-test - Each user's initial domain knowledge was recorded by asking each user to point to the domain object that was called out by the experimenter by its jargon expression.

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 utterances 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 and descriptive expressions, and number of confirmations.

Step 4. Knowledge gain post-test - Each users' knowledge gain during the dialogue task was measured by asking each user to redo the pointing task. The experimenter read out the jargon expression (e.g., "broadband cable") aloud and asked the users to point to the domain entity referred to.

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 (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.

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. Statements about the conversation and the system like, "Conversation with the system was easy," "I would use such a system in future," were answered in a four-point Likert scale on how strongly the user agreed or disagreed with the given statement.

The dialogue corpus was collected from 12 participants; knowledge profiles were acquired from these participants, plus an additional 5 participants reserved for a study of tutorial policy. In total, there were 203 jargon and 202 descriptive expressions used in the dialogues. More statistics are given in Table 3. The participants were students and staff from various backgrounds (arts, humanities, science, medicine, etc.). Every participant was paid £10 after the experiment was finished. Out of the 12 dialogues, 6 used the All-Jargon policy and 6 used the All-Descriptive policy.

Table 3
Corpus statistics (grouped on strategy).

Parameters	Jargon	Descriptive
No. dialogues	6	6
Task completion rate	98.3	98.3
Pre-task score	6.67	8.5
Post-task score	12.33	10.66
Turns	28.17	25.83
Sys words	470.5	471.67
Time (min)	7.7	6.86
Time per turn (sec)	16.49	15.9

5. User Simulations

In this section, we present the second step of our process: building a user simulation. We built a corpus-based user simulation model that simulates the dialogue behavior of a real human user. User simulations are used in place of real users during the training and testing phases of reinforcement learning agents for the following reasons:

1. Training cycles typically require thousands of dialogue episodes to train the agent, and training and testing cycles with real users can be very expensive.
2. Real users could get frustrated with dialogue agents at the initial stage of learning, as they tend to choose random actions that are not adapted to dialogue context.

Several user simulation models have been proposed for use in reinforcement learning of dialogue policies (Georgila, Henderson, and Lemon 2005; Schatzmann et al. 2006, 2007; Ai and Litman 2007). However, they are suited only for learning dialogue management policies, and not for user-modeling policies (i.e., policies to populate the user model and inform other modules of users' domain knowledge). The following user simulation model was therefore designed and implemented to satisfy three requirements: (1) be sensitive to a system's choice of referring expressions, (2) model users' domain knowledge, and (3) learn new expressions during the conversation. Please note that this module is not a part of the actual dialogue system and is used externally in the place of real users. In Sections 6 and 7, we show how the following user simulation was used to train and evaluate the dynamic user modeling behavior of the system.

The user simulation (US) receives the system action $A_{s,t}$ and its referring expression choices $REC_{s,t}$ at each turn as input. Note that the US does not receive as input the natural language utterance from the system. The US responds with a user action $A_{u,t+1}$ (u denoting user) and an environment action $EA_{u,t+1}$. The user action can either be a clarification request (CR) or an instruction response (IR). The user simulation combines three models to simulate the process of a user's understanding of the system's instruction, executing it in the environment, and responding to the system. These three models are for generating clarification requests, environment actions, and instruction responses, as described below.

Clarification request model: This model produces a clarification request CR based on the referent R, type of the referring expression T (i.e., jargon/descriptive), and the current domain knowledge of the user for the referring expression $DK_{u,t}(R, T)$ (i.e., true/false). The referents are classified into "easy" and "hard" in the following way. First, the number of clarification requests per referent entity was calculated from the corpus. Then, those entities whose jargon expressions led to clarification requests more than the mean number of clarification requests were classified as `hard` and others as `easy` entities. For example, *power_adaptor* is `easy` - all users understood this expression; *broadband_filter* is `hard` as there were more than the mean number of clarification requests. The probability of generating a clarification request (CR) for a referring expression depends on the class of the referent $C(R)$, type of the expression used T,

and the user's knowledge of the expression $DK_{u,t}(R,T)$ at time t, and is defined as follows:

$$P(CR_{u,t+1}(R,T)|C(R),T,DK_{u,t}(R,T)) \quad (2)$$

One should note that the actual literal expression was not used in the transaction. Only the entity that it was referring to (R) and its type (T) were used. However, this model simulated the process of interpreting and resolving the expression and identifying the domain entity of interest in the instruction, thereby satisfying our first requirement that the user simulation has to be sensitive to referring expressions used by the system.

Environment action model: An environment action $EA_{u,t}$ was generated using a model based on system dialogue action $A_{s,t}$. This is the probability that the user performed the required action successfully.

$$P(EA_{u,t+1}|A_{s,t}) \quad (3)$$

Instruction response model: An instruction response was generated based on the user's environment action $EA_{u,t+1}$ and the system action $A_{s,t}$. Instruction responses are typical responses to system's instructions and can be either *provide_info*, *acknowledgement*, or *other*. The probability of each of these responses is given by the following model:

$$P(IR_{u,t+1}|EA_{u,t+1},A_{s,t}) \quad (4)$$

The user simulation combined the three models in the following manner. First, it sampled from the clarification request model for each (R,T) in $REC_{s,t}$. If a clarification was produced, it returned it as the user's action (i.e., $A_{u,t+1} = CR_{u,t+1}(R,T)$) and no environment action was produced. If no clarification request was produced, it then sampled from the environment action model (i.e., did the user perform the requested action correctly?) and the instruction response model. The $IR_{u,t+1}$ that was generated was returned to the system as the user action.

All of these models were trained on our corpus data using maximum likelihood estimation and smoothed using a variant of Witten-Bell discounting. The corpus contained 12 dialogues between a non-adaptive dialogue system and real users. According to the data, clarification requests are more likely when jargon expressions are used to refer to the referents that belong to the `hard` class and which the user does not know about. When the system uses expressions that the user knows, the user generally responds to the instruction given by the system.

The trained probabilities are shown in Table 4. Clarification requests occurred only for jargon type expressions and not for descriptive expressions in our corpus. We therefore set the probability of generating one for descriptive expressions to zero.

Using k-means clustering on pre-test knowledge patterns, we created five patterns of users' domain knowledge ($k = 5$). We set k to five so that we obtain three profiles to train with and two additional profiles for testing the learned policy and examining how well it generalizes to the two unseen user types. The models ranged from novices to experts with three intermediate levels, as shown in Table 5. The value T represents that

Table 4
Trained clarification request model (probability of producing a clarification request).

Class (C)	Type (T)	User's Domain Knowledge (DK)	P(CR)
Hard	Jargon	True	5.84
Hard	Jargon	False	84.52
Easy	Jargon	True	2.04
Easy	Jargon	False	15.04

Table 5
Domain knowledge of five different users.

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

a user of the type can identify the referent when the jargon expression is used. The user domain knowledge $DK_{u,t}$ was initially set to one of these models at the start of every conversation. A novice user knew only *power adaptor*, an expert knew all the jargon expressions, and intermediate users knew some of them. We assumed that users can interpret the descriptive expressions for all referents R and resolve their references (i.e., $DK_{u,t}(R, description) = true$). Therefore, they were not explicitly represented. We only coded the user's knowledge of jargon expressions using Boolean variables representing whether the user knew the expression or not. The use of knowledge patterns satisfies the second requirement that the user simulation must model the domain knowledge of the user.

In our corpus of 17 users we had two each of beginners, experts, and int1, four int3 users, and seven int2 users (five users encountered a tutorial policy whose dialogues were not used later on, only their knowledge profiles were used). Corpus data showed that users can learn to associate new jargon expressions with domain entities during the conversation. We modeled this using the **knowledge update model**. This satisfies the third requirement of producing a learning effect and a dialogue behavior that is consistent with an evolving domain knowledge DK_u of the user. 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 for a referent R when the system provided the user with a clarification. This was modeled using the following update rule:

$$\text{if } A_{s,t} == provide\_clarification(R), \text{then } DK_{u,t+1}(R, jargon) = true \quad (5)$$

Users also learned when jargon expressions were repeatedly presented to them. Learning by repetition followed a linear learning relationship (i.e., the greater the number of repetitions, the higher the likelihood of learning), which then converged after a few repetitions. From post-test data, we found that when a jargon expression was given to the user once, the probability that the user learned the association between the term and the entity was 0.55. When it was presented twice or more, the probability was 1. The probability that the user learns a jargon expression is given by a function of the referent (R) and the number of times the jargon expressions are repeated in the conversation, denoted by n, as follows:

$$P(DK_{u,t+1}(R, jargon) = true) = f(R, n) \tag{6}$$

We estimated f as a linear model based on the frequency of each jargon expression and users' post-task recognition scores. Due to the learning effect produced by the system's use of jargon expressions, the final state of the user's domain knowledge ($DK_{u,final}$) may be different from the initial state ($DK_{u,initial}$).

5.1 Evaluation of User Simulation

We measured **dialogue divergence** (DD) based on the Kullback-Leibler (D_{KL}) divergence between real and simulated dialogues to show how realistic our user simulation is. **Kullback-Leibler (KL) divergence**, which is also called **relative entropy**, is a measure of how similar or different two probability distributions are (Kullback and Leibler 1951; Kullback 1959, 1987). Several recent studies have used this metric to evaluate how closely their user simulation models replicate real user behavior (Cuayahuitl et al. 2005; Cuayahuitl 2009; Keizer et al. 2010). Because KL divergence is a non-symmetric measure, DD 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., $D_{KL}(real||simulated)$). DD between two models P and Q is defined as follows:

$$D_{KL}(P||Q) = \sum_{i=1}^{M} p_i * log(\frac{p_i}{q_i}) \tag{7}$$

$$DD(P||Q) = \frac{1}{N} \sum_{i=1}^{N} \frac{D^i_{KL}(P||Q) + D^i_{KL}(Q||P)}{2} \tag{8}$$

The metric measures the divergence between distributions P and Q in N different contexts (i.e., system's dialogue action, entities mentioned, expression type used, and user's knowledge of those expressions) with M responses (i.e., user's dialogue/environment action) per context. Ideally, the dialogue divergence between two similar distributions is close to zero. The divergence of our dialogue action model $P(A_{u,t})$ and the environment action model $P(EA_{u,t})$ with respect to the corpus data were 0.711 and 0.232, respectively. These results were comparable with other recent work on user simulation (Cuayahuitl 2009; Keizer et al. 2010). For a more detailed analysis of our simulation model, see Janarthanam (2011).

6. Learning User-Adaptive Behavior

The final step of our approach is to learn user-adaptive behavior. We used reinforcement learning techniques in order for the system to learn a dynamic user modeling policy. **Reinforcement Learning** (RL) is a set of machine learning techniques in which the learning agent learns the optimal sequence of decisions through trial-and-error learning based on feedback it gets from its environment (Kaelbling, Littman, and Moore 1996; Sutton and Barto 1998). Figure 4 illustrates how a reinforcement learning agent interacts with its environment. The agent is presented with a learning problem in the form of a Markov Decision Process (MDP) consisting of a set of states S, a set of actions A, transition probabilities T from one state to another (when an action is taken), and rewards R associated with such transitions. The agent learns to solve the problem by learning a policy $\pi : s \rightarrow a$ that optimally maps all the states to actions that lead to a high expected cumulative reward. The state of the agent represents the environment as observed by the agent.

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 (Eckert, Levin, and Pieraccini 1997; Levin, Pieraccini, and Eckert 1997; Williams and Young 2003; Cuayahuitl et al. 2005; Henderson, Lemon, and Georgila 2008). Recently, Lemon (2008), Rieser and Lemon (2009), and Dethlefs and Cuayahuitl (2010) have extended this approach to NLG to learn NLG policies to choose the appropriate attributes and strategies in information presentation tasks. However, to our knowledge, the application of RL for dynamically modeling users' domain knowledge and generation of referring expressions based on user's domain knowledge is novel. Figure 5 shows the interaction between the dialogue system and the user simulation (along with environment simulation). The user modeling component (as discussed in Section 3.2) is the learning agent.

The user modeling module was trained using the user simulation presented in Section 5 to learn UM policies that map referring expressions to entities based on the estimated user expertise in the domain. The module was trained in learning mode using the SARSA reinforcement learning algorithm (with linear function approximation) (Shapiro and Langley 2002). The training produced approximately 5,000 dialogues. The user simulation was calibrated to produce three types of users using the Novice, Intermediate (Int2), and Expert profiles from Table 5, randomly but with equal

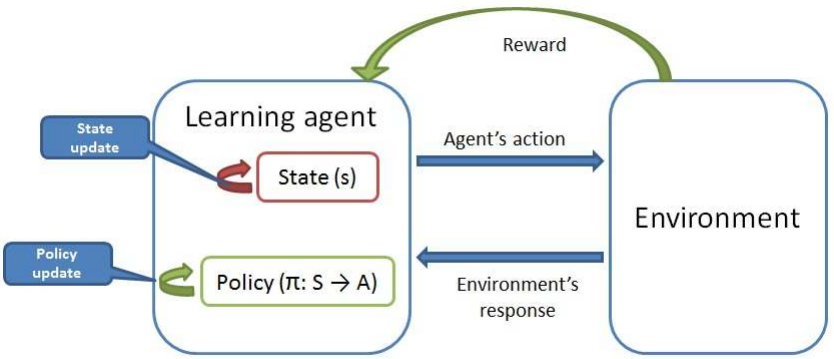

Figure 4
Reinforcement learning.

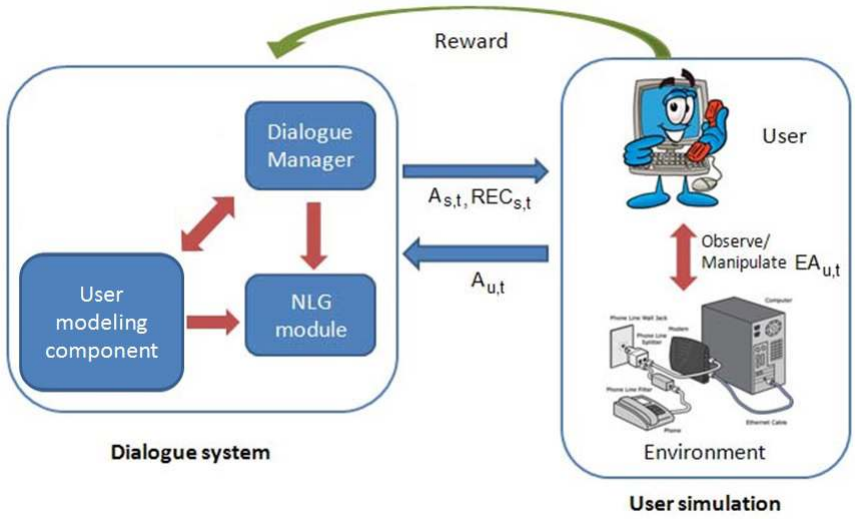

Figure 5
Interaction between the dialogue system and the user simulation (learning).

probability. We did not use all the profiles we had, because we wanted to evaluate how well the learned policy generalizes to unseen intermediate profiles (i.e., Int1 and Int3).

The user modeling state ($UM_{s,t}$) was implemented as follows. It consisted of two variables for each jargon expression x: `user_knows_x` and `user_doesnt_know_x`. They were both initially set to 0. This signified that the agent did not have any information about the user's knowledge of the jargon expression x. The variables were updated using a simple user model update algorithm. If the user responded to an instruction containing the jargon expression x with a clarification request, then `user_doesnt_know_x` was set to 1. On the other hand, if the user responds with an instruction response (*IR*) to the system's instruction, the dialogue manager set `user_knows_x` to 1 and `user_doesnt_know_x` to 0. Each pair of these variables takes only three valid values (out of four possible values); therefore, the state space size for 13 entities is 3^{13} (approximately 1.5 million states).

The actions that were available to the agent were to choose either a jargon expression or a descriptive one for each entity. Once the policy is learned, the decision to choose between using jargon expressions and descriptive expressions for each referent will be made based on the Q-values of the two actions (i.e., *choose_jargon* and *choose_desc*) in the given user model state. The action that gets the highest Q-value will be executed. The Q-value of each action (a) is calculated using the following formula, where s is the user model state with n variables:

$$Q(s,a) = \sum_{i=1}^{n} \theta_a(i) s(i)^T \tag{9}$$

As explained earlier, there are 26 variables (i.e., $n = 26$) in the user model s (s^T is the transpose of s). For each action a, the learning agent learns θ values for each of these variables in the user model ($\theta_a = \theta_a(1), \theta_a(2), .., \theta_a(n)$). Therefore, for each referent, the agent learns two sets of θ values, one for each action. The θ values signify the relevance

of the user's knowledge of various jargon expressions in the domain to its actions. Estimating Q-values as a linear function allows the learning agent to generalize to states not seen during the learning phase (see Section 8.2).

During the learning phase, initially, the θ values are set randomly and the UM policy starts by choosing randomly between the referring expression types for each domain entity in the system utterance, irrespective of the user model state. Once the referring expressions were chosen, the system presented the user simulation with both the dialogue act and referring expression choices. The choice of referring expression affected the user's dialogue behavior. For instance, choosing a jargon expression could evoke a clarification request from the user, based on which the user model state ($UM_{s,t}$) was updated with the new information that the user was ignorant of the particular expression. It should be noted that using a jargon expression is an *information sensing* move that enables the user modeling module to estimate the user's knowledge level. The same process was repeated for every dialogue instruction. At the end of each dialogue, the system was rewarded based on its choices of referring expressions (see Section 6.1). The Q-values of a state-action pair are updated using the following SARSA equation, where α is called the *learning rate* ($0 < \alpha < 1$), which determines how fast or slowly the algorithm learns from its experience, and γ is called the discount factor (Sutton and Barto 1998):

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{10}$$

In addition to choosing actions randomly, the agent can also choose actions based on the Q-values of the state action pair. The former way of choosing actions is called **exploration** and the latter is called **exploitation**. During exploration, the agent tried out new state-action combinations to explore the possibility of greater future rewards. The proportion of exploratory actions were higher at the beginning of the learning phase, but over time it stopped exploring new state-action combinations and used those actions that have high Q-values, which in turn contributed to higher expected reward.

6.1 Reward Function

We wanted the system to learn a policy to present appropriate referring expressions to the user—that is, to present jargon when the user knows it and descriptive otherwise. If the system chose jargon expressions for novice users or descriptive expressions for expert users, penalties were incurred and if the system chose REs appropriately, the reward was high. Although experts might not actively complain about descriptive expressions, they are likely to be less satisfied when the system gives them long instructions instead of using jargon that they can easily handle. Based on the general principle of audience design, the maxim of manner (Gricean maxims of co-operative conversation [Grice 1975]), and principle of sensitivity (Dale 1988), we consider presenting descriptive expressions to experts to be less efficient than using the shorter jargon/expert vocabulary. Although it is not easy to say whether presenting jargon to novices should be weighed the same as presenting descriptive expressions to experts, we use this model as an initial representation for measuring adaptation.

We designed a reward function for the goal of adapting to each user's initial domain knowledge. Our reward function is what we call the "Adaptation Accuracy" score (AA), which calculates how accurately the agent chose the appropriate expressions for each referent in a set of referents (X), with respect to the user's initial knowledge $DK_{u,initial}$.

As before, we use the pair (R, T) to represent a referring expression, where R represents the referent and T represents the type of expression used. So, when the user knew the jargon expression for the referent R, the appropriate expression to use was jargon, and if she or he didn't know the jargon, a descriptive expression was appropriate. This is expressed as function f:

$$f((R,T), DK_{u,initial}) = \begin{Bmatrix} 1 & if\ T = jargon\ and\ DK_{u,initial}(R, jargon) == true \\ 1 & if\ T = desc\ and\ DK_{u,initial}(R, jargon) == false \\ 0 & otherwise \end{Bmatrix} \quad (11)$$

We calculated independent accuracy per referent entity $IA(x)$ and then calculated the overall mean adaptation accuracy (AA) over all referents, as shown in the following. By first calculating independent accuracy for each referent, we ensure that every referent is equally weighted in terms of adaptation when calculating the overall AA.

Where m is the total number of instances of referent R in the conversation with each instance indexed by j, Independent Accuracy (IA) is defined as:

$$IA(R) = \frac{1}{m} \Sigma_{j=1..m} f((R,T)_j, DK_{u,initial}) \quad (12)$$

Where $|X|$ is the total number of distinct domain entities referred to in the conversations, Adaptation Accuracy (AA) is defined as:

$$AA = \frac{1}{|X|} \Sigma_{R \in X}\, IA(R) \quad (13)$$

Other definitions for adaptation accuracy are possible and the automatic optimization would happen in exactly the same way. For instance, it could be defined as adapting to the dynamically changing user's domain knowledge (see Section 8.3). In such a case adaptation accuracy must be calculated based on current domain knowledge of the user ($DK_{u,t}$) instead of the initial domain knowledge ($DK_{u,initial}$).

Another possible metric for optimization would be to weigh each reference *instance* equally, wherein there is no need to calculate Independent Accuracy for each entity and then average them into Adaptation Accuracy, as shown earlier. However, such an approach will lead the learning agent to ignore the entities that are least referred to, and focus on getting the reference to the most frequently referred-to entities right. Investigating other metrics for the reward function is left to future work. In the current set-up, in order to maximize the AA, the system learned to associate the initial state of the user's knowledge with the optimal choice of referring expressions for all the entities equally. We decided to treat each referent equally because the overall task (i.e., setting up a broadband internet connection) would not be successful if even one of the referring expressions fails.

6.2 Learned User Modeling Policy

The user modeling module learned to choose the appropriate referring expressions based on the user model in order to maximize the overall adaptation accuracy, which was our reward function. Figure 6 shows how the agent learned a policy using the data-driven simulation during training. We can see in Figure 6 that towards the end of training the curve plateaus, signifying that learning has converged.

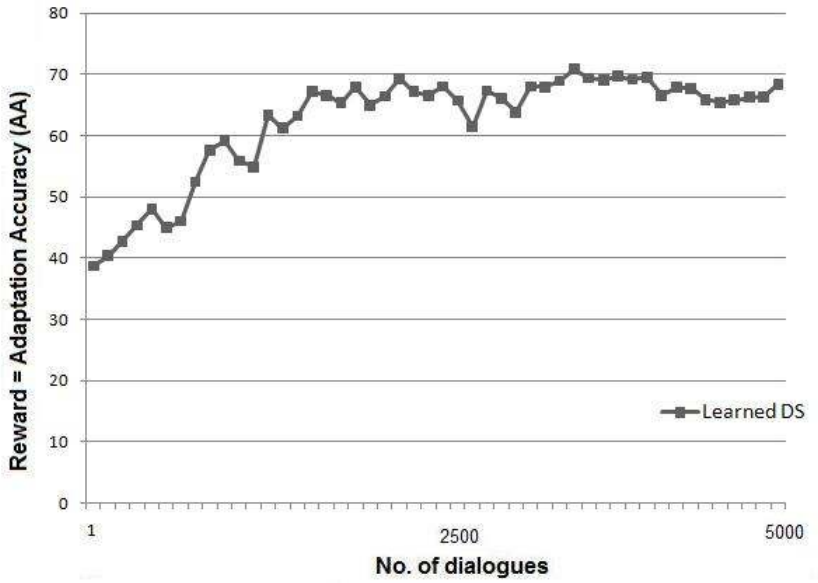

Figure 6
Learning curve: Training.

The system learned a policy to maximize the adaptation accuracy score by quickly sensing the user domain knowledge levels and adapting to this as early as possible. We call this the **Learned-DS** policy as it was learned from interactions with the data-driven user simulation. The system learned that by using jargon expressions, it can discover the user's knowledge about the domain, because users will ask for clarification questions when presented with jargon that they do not know. (Note that this relationship between jargon expressions and information sensing was never explicitly coded into the system.)

Because the agent started the conversation with no knowledge about the user, it learned to use jargon expressions as information sensing moves. Although in the short term this behavior is not rewarding, it allows the system to quickly gather enough information to be able to adapt to the user in order to fetch long-term rewards. For instance, using a jargon expression with a novice user may not be an adaptive move, but it will probably reveal the kind of user that the system was dealing with. Because its goal was to maximize the adaptation accuracy, the agent also learned to restrict such sensing moves and start estimating the user's domain knowledge as soon as possible. By learning to trade off between information-sensing and adaptation, the Learned-DS policy produced high adaptation scores for users with different domain knowledge levels.

It also learned the dependencies between users' knowledge of domain entities as evident in the knowledge profiles (as in Table 5). For instance, when the user asked for clarification on some referring expressions (e.g., *Ethernet cable*), it used descriptive expressions for related domain objects (such as Ethernet light and Ethernet socket). This shows that the system learned the fact that when a user knows *Ethernet cable*, he or she most likely knows *Ethernet light* and *Ethernet socket*. This is evident from the knowledge profiles that (assuming different types of users are equally distributed) there is 0.66 probability that a user knows *Ethernet light* given that he or she knows *Ethernet cable*, and so on. Therefore by sensing the user's knowledge of one entity, it predicts

his or her knowledge of related entities. It also identified a set of non-related entities during the conversation and used this knowledge to sense whenever a new set of non-related entities are introduced in the conversation. For other entities in the same set, it learned to use adaptive choices. Therefore it identified different intermediate users as well. Example dialogues (reconstructed from logged system and user dialogue acts) between real users and the learned policy is given in Appendix A.

7. Evaluation

In this section, we present the details of the evaluation process, the baseline policies, the metrics used, and the results. We evaluated the learned policy and several hand-coded baselines with simulated users and found that the Learned-DS policy produced higher adaptation accuracy than other policies. Another interesting observation is that the evaluation results obtained in simulated environments transfer to evaluations with real users.

7.1 Baseline Policies

In order to compare the performance of the learned policy with hand-coded UM policies, four rule-based adaptive baseline policies were initially developed. We also later developed and evaluated a more advanced baseline (see Section 8.2).

All-Descriptive: Used descriptive expressions for all referents by default.

Jargon-adapt: Used jargon for initial reference for all referents by default, but changed to using descriptive expressions for those referents for which users asked for clarifications. Table 6 provides an example dialogue.

Switching-adapt: This policy started with jargon expressions for initial references and continued using them until the user requested clarification of any entity. After a clarification request, it switched to descriptive expressions for all new referents and continued to use them until the end. Table 7 provides an example dialogue.

Stereotypes: In this policy, we used the knowledge profiles from our data collection. The system started using jargon expressions for the first n turns and then, based on the user's responses, it classified them into one of the five stereotypes (see Table 5) and thereafter used their respective knowledge profiles in order to choose the most

Table 6
Jargon-adapt policy: 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 colorless plastic ends.
Usr: Yes. I have that.
.....
Sys: Please plug one end of the thin black cable with colorless plastic ends into the broadband filter.

Table 7
Switching-adapt policy: 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 colorless 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 colorless plastic ends into
the small white box that has two sockets and a phone plug.

appropriate referring expressions. For instance, if after n turns, the user was classified as a novice, the system used 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 5) and used the one that produced the highest accuracy (i.e., $n = 6$). Note that as the value of n increases from 1, accuracy increases as it provides more evidence for classification. However, after a certain point the adaptation accuracy started to stabilize, because too much sensing is not more informative. Later it started to fall slightly because sensing moves came at the cost of adaptation moves (see Table 8).

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 does not) know a particular expression, this knowledge is exploited in subsequent turns by using the appropriate expressions; and, therefore, the system is adaptive.

We explore additional hand-crafted policies, also using the user profile information, in Section 8.2.

7.2 Additional Evaluation Metrics

We used the adaptation accuracy (see Section 6.1) to measure the level of adaptation to each user. In addition, we also measured other interesting parameters from the conversation (normalized learning gain, dialogue duration, and task completion) to investigate how they are affected by adaptation.

Table 8
Stereotypes: n-values and Adaptation Accuracy (where n is number of turns).

No. of steps	Adaptation Accuracy % (AA)
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

Normalized learning gain (LG): We measured the learning effect on the users using normalized learning gain (*LG*) produced by using unknown jargon expressions. This was calculated using the pre-test (PRE) and post-test (POST) scores for the user domain knowledge (DK_u). Please remember that for simulated runs, the domain knowledge of the user is updated during the interaction using the knowledge update rule. For real users, LG is calculated from their pre- and post-test scores.

$$\textit{Normalized Learning Gain}: \; LG = \frac{POST - PRE}{1 - PRE} \tag{14}$$

Dialogue time (DT): This was the time taken for the user to complete the task. For simulated runs, we estimated the time taken (in minutes) to complete the task using a regression model ($r^2 = 0.98, p = 0.000$) derived from the corpus based on number of words (#(W)), turns (T), and mean user response time (URT).

$$\textit{Dialogue Time}: DT = \frac{19.75 + 0.6 * \#(W) + 0.78 * URT * T}{60} \tag{15}$$

Task completion (TC): This was measured by examining the user's broadband set-up after the task was completed (i.e., the percentage of correct connections that they had made in their final set-up). We used this measure for real users only.

Although our primary objective is to adapt as much as possible to the user, we believe these metrics could be used in future reward functions to achieve goals other than simply adapting to users. For instance, a tutorial dialogue system would aim to optimize on normalized learning gain and would not care much about dialogue time, adaptation, or perhaps even task completion.

7.3 Evaluation with Simulated Users

The user modeling module was operated in evaluation mode to produce 200 dialogues per policy distributed equally over the five user groups (Novice, Int1, Int2, Int3, and Expert). Overall performance of the different policies in terms of Adaptation Accuracy (*AA*), Dialogue Time (*DT*), and Learning Gain (*LG*) are given in Table 9.

Figure 7 shows how the baseline policies as well as the Learned DS policy perform with each user type. It shows that the Learned DS (LDS) policy generalizes well to unseen user types (i.e., Int1 and Int3) and is more consistent than any baseline policy with the different groups, especially for groups Int1 and Int3, whose profiles were not available to the learning agent. This shows that a reinforcement learning agent can learn a policy that generalizes well to unseen user types.

Table 9
Evaluation on five simulated user types.

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 DS	**79.99** (± 10.46)	8.08	0.63

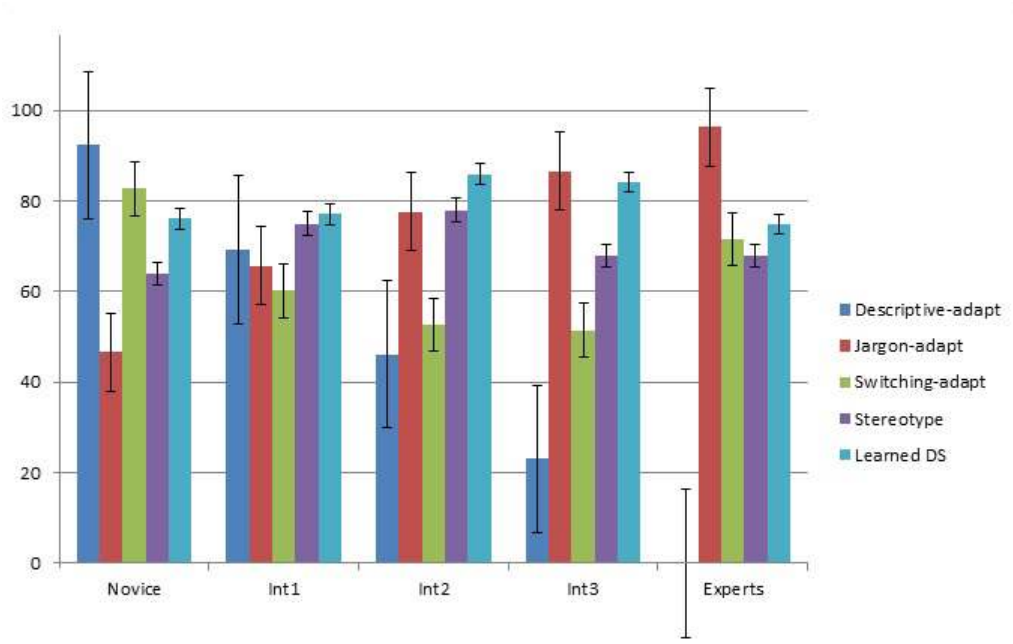

Figure 7
Evaluation: Adaptation Accuracy vs. User types.

In Section 8.2, we further compare the learned policy to additional hand-crafted baseline policies that utilize the user profiles in their adaptation.

A one-way ANOVA was used to test the difference between policies. We found that the policies differed significantly in the adaptation accuracy (AA) metric ($p < 0.0001$). We then used two-tailed paired t-tests (pairing user types) to compare the policies further. We found that the LDS policy was the most accurate (Mean = 79.99, SD = 10.46) in terms of adaptation to each user's initial state of domain knowledge. It outperformed all other policies: 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). Accuracy of adaptation of the LDS 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 ($p = 0.000$, $t = 3.95$, $SE = 23.40$), and Switching-adapt policy ($p = 0.000$, $t = 8.09$, $SE = 22.29$). The LDS policy performed better than the Jargon-adapt policy, because it was able to predict accurately the user's knowledge of referents unseen in the dialogue so far. It performed better than the Stereotype policy because its adaptive behavior takes into account the uncertainty in the user's dialogue behavior. For instance, users did not always ask for clarification when they did not know the jargon expression. They might instead go ahead and do something incorrectly. Therefore, when there is no verbal feedback (i.e., no clarification request) from the user, the system has no information on which a user profile can be picked. However, the learned policy represents this uncertainty in its state transistions and is able to select an appropriate adaptive action. Another point to note is that the LDS policy does not pick a user profile but maps user model states directly to actions, generating either a jargon or descriptive expression for each entity, and so adapts continuously until the end of a dialogue, unlike the stereotype policy, which chooses a profile and sticks with it.

The Jargon-adapt policy performed better than the Switching-adapt and Descriptive policies ($p < 0.05$) in terms of adaptation accuracy. This was because the system can learn more about the user by using more jargon expressions and then using that knowledge to make its later choices more adaptive. Jargon-adapt performed slightly better than the Stereotype policy but the increase in accuracy is not statistically significant ($p = 0.17$). The Stereotype policy also performed significantly better than the Switching-adapt and the Descriptive policies ($p < 0.001$). The Stereotype policy adapted to users globally using their profiles. However, due to uncertainty in user's responses, it was not always possible to pick the right profile for adaptation. This was probably why it outperformed the Switching-adapt and the Descriptive policies and performed as well as the Jargon-adapt policy but did not outperform the Learned-DS policy. The Switching-adapt policy, on the other hand, quickly switched its policy (sometimes erroneously) based on the user's clarification requests but did not adapt appropriately to evidence presented later during the conversation. Sometimes, this policy switched erroneously because of uncertain user behaviors. The Descriptive policy performed very well with novice users but not so with other user types.

In terms of dialogue time (*DT*), the Learned-DS policy was a bit more time-consuming than the Switching-adapt and Descriptive policies but less so than the Jargon-adapt and Stereotype policies. This was because learned policies use sensing moves (giving rise to clarification requests) in order to learn more about the user. The Descriptive policy was non-adaptive and therefore faster than other policies because it only used descriptive expressions and therefore caused no clarification requests from the users. Similarly, due to fewer clarification requests, the Switching-adapt policy also took less dialogue time. Learned policies spent more time in order to learn about the users they interact with before they adapt to them. When the three high-performing policies (by adaptation accuracy) are compared, the Learned-DS policy had the shortest dialogue duration. This was due to better adaptation. The difference between the Learned-DS and the Jargon-adapt policy is statistically significant ($p < 0.05$). However, the difference between the Learned-DS and the Stereotype policy is not significant.

With respect to normalized learning gain (*LG*), the Jargon-adapt policy produced 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 were statistically significant at $p < 0.0001$. The LDS policy 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. Although the use of jargon expressions with novices and intermediates sacrificed adaptation accuracy, it served to increase normalized learning gain as well as populating the user model. Recall that normalized learning gain is not what we aimed to optimize. We merely report this metric as we feel it is interesting to see how adaptation affects learning gain and that this could itself be used as a reward function in the future.

7.4 Evaluation with Real Users

We chose the two best performing policies from our evaluation with simulated users for our final evaluation with real users. Thirty-eight university students from different backgrounds (e.g., Arts, Humanities, Medicine, and Engineering) participated in the evaluation. Seventeen users were given a system with the Jargon-adapt policy and 19 users interacted with a system with the Learned DS (LDS) policy. Data from two other

participants were unusable due to logging issues. Each user was given a pre-task recognition test to record his/her initial domain knowledge. The mean pre-task recognition score of the two groups were tested with Mann-Whitney U test for two independent samples and found to be not significantly different from each other (Jargon-adapt = 7.33, LDS = 7.45). Therefore, there was no bias towards any policy. The experimenter read out a list of technical terms and the user was asked to point out the domain entities laid out in front of them. They were then given one of the two systems, learned or baseline, to interact with. Following the system instructions, they then attempted to set up the broadband connection. When the dialogue had ended, the user was given a post-task test where the recognition test was repeated and their responses were recorded. The user's broadband connection set-up was manually examined for task completion (i.e., the percentage of correct connections that they had made in their final set-up). The user was given the task completion results and was then given a user satisfaction questionnaire to evaluate the features of the system based on the conversation. Example dialogues (reconstructed from logged system and user dialogue acts) between real users and these two policies are given in Appendix A.

All users interacted with a wizarded system using one of the two UM policies. The users' responses were intercepted by a human interpreter (or "wizard") and were immediately annotated as dialogue acts, to which the automated dialogue manager responded with a system dialogue action (the dialogue policy was fixed). The wizards were not aware of the user modeling policy used by the system. The respective policies chose the referring expressions to generate the system utterance for the given dialogue action.

We compare the performance of the two policies on real users using objective parameters and subjective feedback scores. Tests for statistical significance were done using the Mann-Whitney U test for two independent samples (due to the non-parametric nature of the data). Because we measure four metrics, namely, Adaptation Accuracy, Learning Gain, Dialogue Time, and Task Completion Rate, we apply Bonferroni correction and set our α to 0.0125 (i.e., 0.05/4).

Table 10 presents the mean accuracy of adaptation (AA), learning gain (LG), dialogue time (DT), and task completion (TC) produced by the two policies. The LDS policy produced more accurate adaptation than the Jargon-adapt policy ($p = 0.000$, $U = 9.0$, $r = -0.81$). The use of the LDS policy resulted in less dialogue time ($U = 73.0$, $p = 0.008$, $r = -0.46$) and higher task completion ($U = 47.5$, $p = 0.0006$, $r = -0.72$) than the Jargon-adapt policy. However, there was no significant difference in LG. Another important point to note is that the order of ranking in terms of adaptation accuracy from the simulated user evaluation is preserved in the real user evaluation as well: LDS policy scores better than Jargon-adapt policy in terms of AA both with simulated and real users. We tested for correlation between the above metrics using Spearman's rho

Table 10
Evaluation with real users.

	Jargon-adapt	Learned DS	Sig.
Adaptation Accuracy (%)	63.91 ($\pm$ 8.4)	84.72 ($\pm$ 4.72)	*
Learning Gain	0.71 ($\pm$ 0.26)	0.74 ($\pm$ 0.22)	
Dialogue Time (mins)	7.86 ($\pm$ 0.77)	6.98 ($\pm$ 0.93)	*
Task Completion Rate (%)	84.7 ($\pm$ 14.63)	99.47 ($\pm$ 2.29)	*

*Statistical significance ($p < 0.0125$).

Table 11
Real user feedback.

	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

correlation. We also found that AA correlates positively with task completion rate (TCR) ($r = 0.584$, $p = 0.000$) and negatively with DT ($r = -0.546$, $p = 0.001$). 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.

Table 11 presents how the users subjectively scored different features of the system on an agreement scale of 1 to 4 (with 1 = strongly disagree and 4 = strongly agree), based on their conversations with the two different strategies. The difference in overall satisfaction score, calculated as the mean of all the questions Q1 to Q8 (with Q2 reversed), was not significant (Jargon $= 3.1 \pm 0.38$, Learned $= 3.35 \pm 0.32$, $p = 0.058$). Although there is statistical difference between the policies in the objective metrics, there is no significant difference between them in any of the user ratings.

Users seemed unable to recognize the nuances in the way the system adapted to them (Q3) and they did not rate the Learned-DS policy any higher than the Jargon-adapt policy regarding whether it was easy to identify objects (Q4). They could have been satisfied with the fact that both the systems adapted 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 the system well in terms of future use (Q8), dialogue time (Q5), and ease of conversation (Q7); but in common with experiments in dialogue management (Lemon, Georgila, and Henderson 2006), it seems that users find it difficult to evaluate these improvements subjectively. 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 was better. Results in Table 11 lead us to conclude that perhaps users need to directly compare two or more strategies in order to better judge the differences between strategies, or perhaps the differences are just too subtle for users to notice. Another point to note is that the participants, although real humans, were performing the task in a laboratory setting and not in a real setting (e.g., at home where they are setting up their own home broadband connection).

8. Discussion

8.1 Application of Our Approach

Our approach could be generally useful in dialogue systems where users' domain knowledge influences the conversations between users and the system. Some systems will simply aim to adapt to the user as much as possible and do not need to attend to users' learning, which is the approach we have taken in this article. For instance, a city

navigation system that interacts with locals and tourists (such as Rogers, Fiechter, and Thompson 2000; Janarthanam et al. 2013) should use proper names and descriptions of landmarks appropriately to different users to guide them around the city. A technical support system helping expert and novice users (such as Boye 2007) should use referring expressions and instructions appropriate to the user's expertise. An Ambient Intelligence Environment in a public space (e.g., museum) interacting with visitors (such as Lopez-Cozar et al. 2005) can guide visitors and describe the exhibits in a language that the user would appreciate and understand.

8.2 Comparison with More Intelligent Hand-Coded Policies

Although some of our hand-coded policies adapted to users, most of them did not use internal user models (except the Stereotype policy). We therefore also compared the performance of our learned policy with a more intelligent hand-coded policy that uses all the five user knowledge profiles: "Active_Stereotype_5Profiles" (AS5). The AS5 policy made use of the knowledge profiles just like the Stereotype policy described in Section 7.1. However, the difference was that this policy used the stereotype information to actively select one of the five possible stereotypes to apply from the start of the conversation (unlike the Stereotype policy, which waited until six turns to make a decision). This is done through a process of elimination. Initially, all five stereotype profiles are considered possible. The policy starts the conversation with jargon expressions, and as evidence is gathered about the user's knowledge of the jargon expressions, it eliminates those profiles that are incompatible with the evidence. For instance, if the user knows the expression *Livebox*, the policy eliminates *beginner* profile from the list of possibilities. It goes on until it has narrowed down the possibilities to one profile in a similar fashion. The last remaining profile was then used for adapting to the user. During this process of elimination, it also continuously estimates the user's domain knowledge based on the stereotypes that are still under consideration. This is done so that if all profiles under consideration indicate that the user does not know a particular jargon expression, a descriptive expression can be used instead to improve adaptation. Otherwise, the policy used jargon expressions as information sensing moves. This policy was run to produce 200 dialogues with the user simulation (see Section 5). The user simulation generated the behavior of all five types of user with equal probability. The average AA of the AS5 policy was 77.52% ($\pm$23.36). We found no significant difference between the means of the AS5 policy and the LDS policy using a paired t-test (pairing user types). How these two policies compare for each user type can be seen in Figure 8. Whereas there was no significant difference in means for Int2, Int3, and Expert users, for Beginners, AS5 was better than LDS (AS5 = 83.72, LDS = 76.92, $p = 0.009$) and for Int1, LDS was better (LDS = 77.03, AS5 = 65.72, $p = 0.0001$). Although it may seem that the learned policy is only as good as a smart hand-coded policy, it must be noted that the AS5 policy uses five user profiles and the LDS policy was trained using only three profiles.

It therefore seems reasonable to compare the LDS policy with a version of the active stereotype policy that only uses the same three user profiles (Beginner, Int2, and Expert) that the learned policy had access to during training. We call this policy "Active_Stereotype_3Profiles" (AS3). It works the same way as the AS5 policy but only has three profiles to start with. We ran this policy with the user simulation and compared the adaptation accuracy produced to the LDS policy. The overall average adaptation accuracy over all user types for the AS3 policy was 66.98 ($\pm$25.79). This

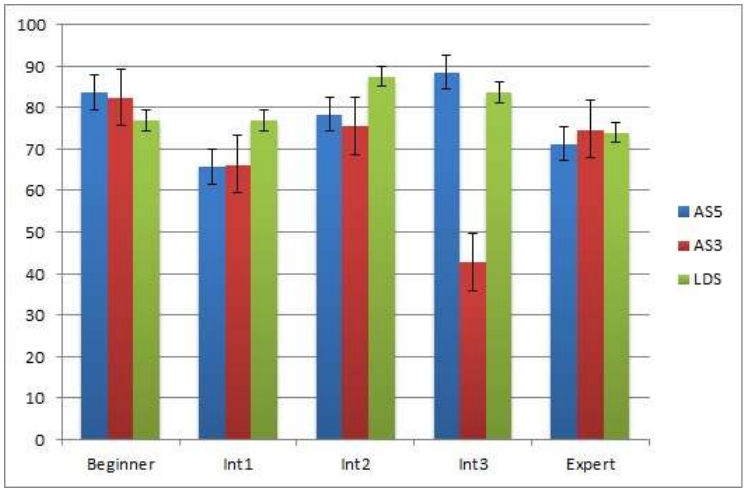

Figure 8
Evaluation - Adaptation Accuracy vs. User types (LDS vs. AS policies).

was significantly lower than the LDS policy ($p = 0.0001$). We also compared the two policies per user type (see Figure 8). The AS3 policy was better than the LDS policy for Beginners (LDS = 76.92, AS3 = 82.49, $p = 0.02$), no statistical difference was found for Int2 and Experts, and the LDS policy was better than AS3 for Int1 (LDS = 77.03, AS3 = 66.34, $p = 0.0001$) and Int3 users (LDS = 83.64, AS3 = 42.84, $p = 0.0001$). This shows that the LDS policy is able to generalize well to unseen users (i.e., Int1 and Int3), better than a smart hand-coded policy that had the same knowledge of user profiles.

8.3 Learning to Adapt to a Dynamically Changing User Knowledge

In reality, users often learn during a technical conversation. This is how we modeled users in our user simulation. However, we only learned a policy that adapts to the initial state of the user's knowledge. We see this as a first step towards learning a more complex policy that will adapt to a dynamically changing user knowledge state. Adapting to dynamically changing user knowledge requires additional representation in the system's user model regarding what users might learn during the conversation, in addition to what they already know. Furthermore, the system will have to model the nuances between expressions that are easy to learn and those that are harder to learn, and also that users' learning might be affected by how many times an entity is repeatedly referred to in a conversation. The system may also need to model the process of users forgetting recently learned expressions, especially in long conversations involving many domain entities. There are several applications of this approach to user modeling. For instance, an assistive health care system that interacts with patients to educate and assist them in taking care of themselves (Bickmore and Giorgino 2004) should be able to adapt to patients' initial levels of knowledge and in subsequent dialogues change its language according to the improvement in the patient's understanding and improving knowledge of the domain. Similarly, a tutorial dialogue system that tutors students or trains personnel in industry (such as Dzikovska et al. 2007) should adapt to the needs of the learner in terms of their levels of understanding and expertise. Such systems pay attention to learning gain, but aim to keep the

user in the zone of proximal development (Vygotsky 1978) and therefore not use too many jargon expressions or complex instructions. We leave these issues for future research.

8.4 Extending Our Approach to Content Selection and Dialogue Management

In this section, we discuss how our approach to user modeling (that we have demonstrated in choosing referring expressions) can be extended to content selection and dialogue management. In the current set-up, for every instruction the dialogue manager chooses to give the user, the NLG module uses the corresponding template. In other words, there is a one-to-one mapping between the dialogue move and the content of the system utterance; and there is no choice for the NLG module to make in terms of choosing the content of the utterance (except for choosing the referring expressions). Suppose there is more than one choice for every dialogue move, such as a *detailed* instruction that anyone can process and an *abstract* one that only experts can handle (see Table 12, for example). The choice between the two can be made based on a user modeling policy similar to the one we used for choosing referring expressions. This may require modeling user's domain knowledge not only in terms of their ability to identify domain objects using their technical names but also whether they understand domain specific verbs (e.g., *connect*) and know domain-specific procedures, or are capable of domain-specific planning and execution given a high-level instruction. For instance, for the instruction, "Connect the modem to your computer using the Ethernet cable," the user must be able to not only identify the Ethernet socket, but must also be able to plan or know two low-level moves: connecting the Ethernet cable to the modem and connecting the other end of the cable to the computer, which in turn requires him/her to know where (i.e., which socket and its location) in the modem and the computer the Ethernet cable should be plugged in. This knowledge about the user needs to be modeled in the user modeling state space.

Similarly, we can extend the user modeling policy to support dialogue management. Dialogue management is the process of maintaining the dialogue context, and based on that choosing an appropriate dialogue action at an appropriate time. In this study, we presented a simple dialogue manager that uses a finite state machine to present a sequence of detailed instructions to the user. However, it is reasonable to sometimes group a smaller sequence of instructions and refer to such a procedure by a technical name. For instance, what can be referred to as *ping 192.168.0.1* can be broken down into a sequence of instructions such as clicking on the *Start* button, searching for and opening the command line, and typing *ping 192.168.0.1*. Although some users know how to plan this sequence of actions given the technical action term *ping*, others may need to be given step-by-step guidance (see Figure 9). The dialogue manager can

Table 12
Detailed vs. abstract instruction.

Detailed instruction
Sys: Please plug one end of the Ethernet cable into the Ethernet socket of the modem and plug the other end of the Ethernet cable into the Ethernet socket of your computer.
Abstract instruction
Sys: Connect the modem to your computer using the Ethernet cable.

Instruct to carry out a domain procedure
Sys: *Ping 192.168.0.1* to see if your modem is reachable.

Instruct to carry out a sequence of actions step by step
Sys: Open command prompt.
Usr: What is a command line?
Sys: Never mind. Click on the *Start* button.
Usr: Ok.
Sys: Search for command prompt.
Usr: Ok. Found it.
Sys: Execute command prompt.
Usr: Ok.
Sys: Type *ping 192.168.0.1*.

Figure 9
Domain procedure vs. step-by-step instruction.

decide between the two: Give procedure name with parameters or give detailed step-by-step instructions, based on the user model. Just like content selection, this needs the domain knowledge of the user to be modeled at a detailed level, such as knowledge of domain specific procedures (e.g., pinging). Another interesting domain where user modeling can help dialogue management is pedestrian navigation. Janarthanam et al. (2013) use a similar approach where users who are locals may know how to get to most well-known streets but not necessarily to their destinations. In most cases, users may want to go to a destination, but may know how to navigate part-way by themselves. In such cases, users could be asked to navigate to the well-known streets by themselves and then navigate using turn-by-turn instructions until they reach their destination. However, tourists may need step-by-step instructions right from the beginning. By dynamically modeling users' knowledge of the city, the dialogue manager can decide whether to ask the user to self-navigate part-route, or ask him or her if he or she can do that, or start navigating him or her right away. When adaptation happens at the level of content selection or dialogue management, we argue that adaptive systems such as ours can choose instructions of appropriate complexity and therefore save more dialogue time than non-adaptive competitors such as Descriptive and Switching-adapt policies.

Another way in which the dialogue manager can be involved in user modeling is to explicitly question the user about the parameters modeled in the user model. Instead of just using a jargon expression to sense if the user knows it or not, the system will now have a choice to ask a probing question, such as "Do you know what a broadband filter is?" or "Do you know how to connect the modem to your computer using a Ethernet cable?" (as in Cawsey [1993] but learned from interactions with a user simulation). The dialogue manager ideally has to choose between the two approaches to sense information from the user or balance between them based on a long-term reward. This reward may not be based just on adaptation accuracy but also other factors such as time taken to complete the task, task success, user satisfaction, and so on. Here we see an interesting trade-off between explicitly and implicitly sensing information from the users about their domain knowledge. It should also be noted that evidence of users' knowledge at one level could be used at other levels and should be modeled by the system. For instance, if the system discovers that the user does not know *Ethernet cable*, this evidence can be used in content selection and dialogue management levels as well in selecting appropriate actions.

8.5 Advantages of the Learning Approach

To summarize, the Reinforcement Learning approach to learning user modeling policies has the following advantages compared with other approaches that could deliver similar results:

Automatic optimization: The policy is learned guided by the reward function. One should note that by modifying the reward function, the agent could be made to learn a different policy automatically. In the given set-up, we could set LG instead of AA as reward function and the agent will learn a policy to maximize LG and not care for adaptation. More complex policies can be learned when more than one metric is combined (e.g., in the form of a weighted linear function) in the reward function to balance different goals of the system (e.g., learning gain, lexical alignment). In such cases, the agent learns a policy that trades off between the different metrics optimally. For example, Rieser and Lemon (2011) have shown how a reinforcement learning agent can learn a dialogue management policy to balance different metrics such as task success and dialogue duration to optimize the overall user satisfaction score.

Generalization: We have shown that the learned policy generalizes well to unseen users. We have also shown that even a smart hand-coded policy utilizing user profiles could not handle unseen users very effectively in comparison with the learned policy.

Learning from a small corpus: By factoring the user simulation into three models (Clarification request, Environment, and Instruction Response), we were able to learn a realistic user simulation from a small corpus of 12 dialogues and 17 user profiles.

Learning on-line: A recent study by Gašić et al. (2013) has shown that it is possible to learn such user modeling policies online by interacting with real users.

9. Conclusion

In this article we presented a novel sense-predict-adapt approach to dynamic user modeling and adaptation in dialogue systems. In this approach, the system learned to choose appropriate referring expressions to refer to domain objects based on users' domain knowledge levels using an RL framework and data-driven user simulations. It learned an adaptive policy by interacting with simulated users with different levels of domain knowledge from experts to novices, based on a small amount of training data. It learned to trade off between adaptive moves and information sensing moves automatically to maximize the overall adaptation accuracy. The learned policy started the conversation with information sensing moves, learned a little about the user, and started adapting dynamically as the conversation progressed. It also learned the dependencies between entities in the users' domain knowledge, which it uses effectively for predicting users' knowledge of future entities. Although we focused on adaptively choosing referring expressions in technical instructions, we discussed how our model can be extended to other levels of adaptation as well such as content selection and aggregation. We discussed how this approach can be applied to spoken dialogue systems in domains such as technical support and tutoring, when users' domain knowledge can be dynamically modeled and adapted to.

We presented a novel user simulation module that can be trained from a small corpus of non-adaptive dialogues and simulate users' dialogue behavior in a technical conversation. It simulated users with different levels of domain knowledge. We trained the system using the user simulation and evaluated the learned policy along with several other hand-coded policies. In an evaluation with simulated users, we showed that the learned policy generalizes to unseen users and outperforms several different baseline policies, including a policy that makes use of user domain knowledge profiles from the corpus. In our real user evaluation study, we showed that the learned policy performs better than a simple adaptive hand-coded policy with real users in terms of accuracy of adaptation, dialogue time, and successful task completion. This article also provides further evidence that evaluation results obtained in simulated environments can transfer reliably to evaluations with real users (Lemon, Georgila, and Henderson 2006). Finally, we also compared the learned policy to a variety of carefully hand-crafted adaptive policies that utilize the user knowledge profiles to adapt their choices of referring expressions throughout a conversation. We show that the learned policy generalizes better to unseen user profiles than these hand-coded policies, while having comparable performance on known user profiles.

Whether the learned policy would perform better than a hand-coded policy that was painstakingly crafted by a domain expert (or learned using supervised methods from an expert-layperson corpus) is an interesting question that needs further exploration. Also, it would also be interesting to make the learned policy account for the user's learning behavior and adapt accordingly.

Appendix A: Examples of Dialogue with Real Users

An example dialogue between a real user and the jargon-adapt policy
(System choices shown in **boldface**.)

(1) Sys: Are you ready?
(2) Usr: Yes.
(3) Sys: Do you have a **livebox** in the package?
(4) Usr: Yes.
(5) Sys: Do you have a **broadband cable** in the package?
(6) Usr: Which one is the broadband cable?
(7) Sys: The broadband cable is the thin black cable with colorless plastic ends.
(8) Usr: Yes.
(9) Sys: Do you have an **Ethernet cable** in the package?
(10) Usr: Yes.
(11) Sys: Do you have a **power adaptor** in the package?
(12) Usr: Yes.
(13) Sys: Do you have a **broadband filter** in the package?
(14) Usr: Which one is the broadband filter?
(15) Sys: The broadband filter is the small white box that has two sockets and a phone plug.
(16) Usr: Yes.
(17) Sys: Good. We seem to have everything we need. Now follow my instructions to set up the connections. Ready?
(18) Usr: I am ready.
(19) Sys: Please disconnect the phone from the **phone socket** on the wall.

(20) Usr: Done.
(21) Sys: Please take the **power adaptor**.
(22) Usr: Yes.
(23) Sys: Please plug the **power adaptor** into the two-pin mains power socket.
(24) Usr: Yes.
(25) 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**.
(26) Usr: Yes.
(27) Sys: Did all the lights on the front panel of the **livebox** turn on and go off after a few seconds.
(28) Usr: Yes.
(29) Sys: Place the **livebox** with the Orange label facing up. What is the status of the **power light** on the **livebox**?
(30) Usr: Its flashing red.
(31) Sys: Please plug one end of the **thin black cable with colorless plastic ends** into the **ADSL socket** on the **livebox**.
(32) Usr: Yes.
(33) Sys: Take the **small white box that has two sockets and a phone plug**.
(34) Usr: Yes.
(35) Sys: Please plug the other end of the **thin black cable with colorless plastic ends** into the **small white box that has two sockets and a phone plug**.
(36) Usr: Yes.
(37) Sys: Please plug the **small white box that has two sockets and a phone plug** into the **phone socket** on the wall.
(38) Usr: Yes.
(39) Sys: What is the status of the **broadband light** on the **livebox**?
(40) Usr: Its flashing red.
(41) Sys: Please plug one end of the **Ethernet cable** into the **Ethernet socket** on the **livebox**.
(42) Usr: Yes.
(43) Sys: Please connect the other end of the **Ethernet cable** into the **Ethernet socket** on the back panel of your computer.
(44) Usr: Yes.
(45) Sys: What is the status of the **Ethernet light** on the **livebox**?
(46) Usr: Its red.
(47) 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.
(48) Usr: Ok.
(49) Sys: Good. We have now finished setting up your broadband Internet connection.

In this dialogue, the Jargon-adapt policy learns in turns (6) and (14) that the user does not know *broadband cable* and *broadband filter*. It uses this information to adapt to the user by using descriptive expressions for those entities later in turns (31), (33), (35), and (37). However, it does not use this information to adapt with respect to other entities referred to in the conversation.

An example dialogue between a real user and the Learned-DS policy
(System choices shown in **boldface**.)

(1) Usr: Yes.
(2) Sys: Do you have a **livebox** in the package?
(3) Usr: Yes.
(4) Sys: Do you have a **broadband cable** in the package?
(5) Usr: Which one is the broadband cable?
(6) Sys: The broadband cable is the thin black cable with colorless plastic ends.
(7) Usr: Yes.
(8) Sys: Do you have an **Ethernet cable** in the package?
(9) Usr: Yes.
(10) Sys: Do you have a **power adaptor** in the package?
(11) Usr: Yes.
(12) Sys: Do you have a **broadband filter** in the package?
(13) Usr: Which one is the broadband filter?
(14) Sys: The broadband filter is the small white box that has two sockets and a phone plug.
(15) Usr: Yes.
(16) Sys: Good. We seem to have everything we need. Now follow my instructions to set up the connections. Ready?
(17) Usr: I am ready.
(18) Sys: Please disconnect the phone from the **phone socket** on the wall.
(19) Usr: Done.
(20) Sys: Please take the **power adaptor**.
(21) Usr: Yes.
(22) Sys: Please plug the **power adaptor** into the two-pin mains power socket.
(23) Usr: Yes.
(24) 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**.
(25) Usr: Yes.
(26) Sys: Did all the lights on the front panel of the **livebox** turn on and go off after a few seconds.
(27) Usr: Yes.
(28) Sys: Place the **livebox** with the Orange label facing up. What is the status of the **power light** on the **livebox**?
(29) Usr: Its flashing red.
(30) Sys: Please plug one end of the **thin black cable with colorless plastic ends** into the **ADSL socket** on the **livebox**.
(31) Usr: Yes.
(32) Sys: Take the **small white box that has two sockets and a phone plug**.
(33) Usr: Yes.
(34) Sys: Please plug the other end of the **thin black cable with colorless plastic ends** into the **small white box that has two sockets and a phone plug**.
(35) Usr: Yes.
(36) Sys: Please plug the **small white box that has two sockets and a phone plug** into the **phone socket** on the wall.
(37) Usr: Yes.
(38) Sys: What is the status of the **second light from the left** on the **livebox**?
(39) Usr: Its flashing red.

(40) Sys: Please plug one end of the **Ethernet cable** into the **square socket with red stripe** on the **livebox**.
(41) Usr: Yes.
(42) Sys: Please connect the other end of the **Ethernet cable** into the **Ethernet socket** on the back panel of your computer.
(43) Usr: Yes.
(44) Sys: What is the status of the **fourth light from the left** on the **livebox**?
(45) Usr: Its red.
(46) 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.
(47) Usr: Ok.
(48) Sys: Good. We have now finished setting up your broadband Internet connection.

In this dialogue, the Learned-DS policy adapts to a user who does not know the expressions *broadband filter* and *broadband cable*. It uses this information to adapt by using descriptive expressions for those entities later in the conversation just like the Jargon-adapt policy. In addition, it also uses descriptive expressions for two other entities: Ethernet light and broadband light (see turns (38) and (44)).

Acknowledgments
The research leading to these results has received funding from British Council (UKIERI PhD scholarships 2007–2010), the European Community's Seventh Framework Programme (FP7/2007–2013) under grant agreements no. 216594 (CLASSiC project `www.classic-project.org`) and 270019 (SPACEBOOK project `www.spacebook-project.eu`), and from the EPSRC (project no. EP/G069840/1).

References

Ai, H. and D. Litman. 2007. Knowledge consistent user simulations for dialog systems. In *Proceedings of Interspeech 2007*, pages 2,697–2,700, Antwerp.
Akiba, T. and H. Tanaka. 1994. A Bayesian approach for user modeling in dialogue systems. In *Proceedings of the 15th Conference on Computational Linguistics - Volume 2*, Kyoto.
Bell, A. 1984. Language style as audience design. *Language in Society*, 13(2):145–204.
Belz, A. and S. Varges. 2007. Generation of repeated references to discourse entities. In *Proceedings ENLG-2007*, pages 8–16, Schloss Dagstuhl.
Bickmore, T. and T. Giorgino. 2004. Some novel aspects of health communication from a dialogue systems perspective. In *AAAI Fall Symposium on Dialogue Systems for Health Communication*, pages 275–291, Washington, DC.
Boye, J. 2007. Dialogue management for automatic troubleshooting and other problem-solving applications. In *Proceedings SIGDial'07*, pages 247–255, Antwerp.
Cawsey, A. 1993. User modeling in interactive explanations. *User Modeling and User-Adapted Interaction*, 3(3):221–247.
Cheng, H., H. Bratt, R. Mishra, E. Shriberg, S. Upson, J. Chen, F. Weng, S. Peters, L. Cavedon, and J. Niekrasz. 2004. A Wizard of Oz framework for collecting spoken human-computer dialogs. In *Proceedings of the International Conference on Spoken Language Processing*, pages 2,269–2,272, Jeju.
Clark, H. H. and G. L. Murphy. 1982. Audience design in meaning and reference. In J. F. Leny and W. Kintsch, editors, *Language and Comprehension*. North-Holland Publishing Company, Amsterdam.
Cuayahuitl, H. 2009. *Hierarchical Reinforcement Learning for Spoken Dialogue Systems*. Ph.D. thesis, University of Edinburgh.
Cuayahuitl, H., S. Renals, O. Lemon, and H. Shimodaira. 2005. Human--computer dialogue simulation using hidden Markov models. In *Proceedings of ASRU 2005*, Cancun.
Dale, R. 1988. *Generating Referring Expressions in a Domain of Objects and Processes*. Ph.D. thesis, University of Edinburgh.
Demberg, Vera, Andi Winterboer, and Johanna D. Moore. 2011. A strategy for information presentation in spoken dialog systems. *Computational Linguistics*, 37(3):489–539.

Dethlefs, N. and H. Cuayahuitl. 2010. Hierarchical reinforcement learning for adaptive text generation. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 37–45, Dublin.

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

Eckert, W., E. Levin, and R. Pieraccini. 1997. User modeling for spoken dialogue system evaluation. In *Proceedings of ASRU 1997*, pages 80–87, Santa Barbara, CA.

Forbes-Riley, K. and D. Litman. 2010. Designing and evaluating a wizarded uncertainty-adaptive spoken dialogue tutoring system. *Computer Speech and Language*, 25(1):105–126.

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

Gašić, M., C. Breslin, M. Henderson, D. Kim, M. Szummer, B. Thomson, P. Tsiakoulis, and S. Young. 2013. On-line policy optimisation of Bayesian spoken dialogue systems via human interaction. In *Proceedings of ICASSP 2013*, Canada.

Gatt, A. and A. Belz. 2008. Attribute selection for referring expression generation: New algorithms and evaluation methods. In *Proceedings of INLG-2008*, pages 50–58, Salt Fork, OH.

Gatt, A. and K. van Deemter. 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)*, Amsterdam.

Georgila, K., J. Henderson, and O. Lemon. 2005. Learning user simulations for information state update dialogue systems. In *Proceedings of Eurospeech/ Interspeech*, pages 893–896, Lisbon.

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

Hajdinjak, M. and F. Miheli. 2003. The Wizard of Oz system for weather information retrieval. In *Proceedings of the 6th International Conference TSD*, pages 400–405, Czech Republic.

Henderson, J., O. Lemon, and K. Georgila. 2008. Hybrid reinforcement/supervised learning of dialogue policies from fixed datasets. *Computational Linguistics*, 34(4):487–512.

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.

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

Janarthanam, S. 2011. *Learning User Modeling Strategies for Adaptive Referring Expression Generation in Spoken Dialogue Systems*. Ph.D. thesis, University of Edinburgh.

Janarthanam, S. and O. Lemon. 2009. A Wizard-of-Oz environment to study referring expression generation in a situated spoken dialogue task. In *Proceedings ENLG'09*, pages 94–97, Athens.

Janarthanam, S., O. Lemon, P. Bartie, T. Dalmas, A. Dickinson, X. Liu, J. Mackaness, and W. Goetze. 2013. Evaluating a city exploration dialogue system with integrated question-answering and pedestrian navigation. In *Proceedings of ACL 2013*, pages 1,660–1,668.

Kaelbling, L. P., M. L. Littman, and A. W. Moore. 1996. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285.

Kass, R. 1991. Building a user model implicitly from a cooperative advisory dialogue. *User Modeling and User-Adapted Interaction*, 1:203–258.

Keizer, S., M. Gašić, F. Jurcicek, F. Mairesse, B. Thomson, K. Yu, and S. Young. 2010. Parameter estimation for agenda-based user simulation. In *Proceedings of SIGDial 2010*, pages 116–123, Tokyo.

Kittredge, R., T. Korelsky, and O. Rambow. 1991. On the need for domain communication knowledge. *Computational Intelligence*, 7(4):305–314.

Kobsa, A. and W. Wahlster. 1989. *User Models in Dialog Systems*. Springer Verlag, Berlin.

Krahmer, E. and M. Theune. 2002. Efficient context-sensitive generation of referring expressions. In K. van Deemter and R. Kibble, editors, *Information Sharing: Reference and Presupposition in Language Generation and Interpretation*. CSLI, Stanford, CA, pages 223–264.

Krahmer, E., S. van Erk, and A. Verleg. 2003. Graph-based generation of referring expressions. *Computational Linguistics*, 29(1):53–72.

Kullback, S. 1959. *Information Theory and Statistics*. John Wiley and Sons, New York.

Kullback, S. 1987. Letter to the Editor: The Kullback-Leibler distance. *The American Statistician*, 41(4):340–341.
Kullback, S. and R. A. Leibler. 1951. On Information and Sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86.
Lemon, O. 2008. Adaptive natural language generation in dialogue using reinforcement learning. In *Proceedings of SEMdial'08*, pages 149–156, London.
Lemon, O., K. Georgila, and J. Henderson. 2006. Evaluating effectiveness and portability of reinforcement learned dialogue strategies with real users: The TALK TownInfo evaluation. In *Proceedings of IEEE/ACL Spoken Language Technology 2006*, pages 178–181, Palm Beach.
Levin, E., R. Pieraccini, and W. Eckert. 1997. Learning dialogue strategies within the Markov decision process framework. In *Proceedings of ASRU 1997*, pages 72–79, Santa Barbara, CA.
Lopez-Cozar, R., Z. Callejas, M. Gea, and G. Montoro. 2005. Multimodal, multilingual and adaptive dialogue system for ubiquitous interaction in educational space. In *Proceedings of Applied Spoken Language Interaction in Distributed Environments (ASIDE)*, Aalborg.
Mairesse, F. and M. Walker. 2010. Towards personality-based user adaptation: Psychologically informed stylistic language generation. *User Modeling and User-Adapted Interaction*, 20:3:227–278.
McKeown, K., J. Robin, and M. Tanenblatt. 1993. Tailoring lexical choice to the user's vocabulary in multimedia explanation generation. In *Proceedings ACL 1993*, pages 226–234, Columbus, OH.
Nguyen, L. and P. Do. 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, Baton Rouge, LA.
Paris, C. L. 1984. Determining the level of expertise. In *First Annual Workshop on Theoretical Issues in Conceptual Information Processing*, Atlanta, GA.
Rambow, O. 1990. Domain communication knowledge. In *Proceedings of the Fifth International Workshop on Natural Language Generation 1990*, pages 87–94, Dawson, PA.
Reiter, E. and R. Dale. 1992. A fast algorithm for the generation of referring expressions. In *Proceedings COLING-1992*, pages 232–238, Nantes.
Reiter, E. and R. Dale. 1995. Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science*, 18:233–263.
Reiter, E. and R. Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press.
Rieser, V. and O. Lemon. 2009. Natural language generation as planning under uncertainty for spoken dialogue systems. In *Proceedings EACL 2009*, pages 638–691, Athens.
Rieser, V. and O. Lemon. 2011. Learning and evaluation of dialogue strategies for new applications: Empirical methods for optimization from small data sets. *Computational Linguistics*, 37:1:153–196.
Rogers, S., C. Fiechter, and C. Thompson. 2000. Adaptive user interfaces for automotive environments. In *IEEE Intelligent Vehicles Symposium*, pages 662–667, Dearborn, MI.
Schatzmann, J., B. Thomson, K. Weilhammer, H. Ye, and S. J. Young. 2007. Agenda-based user simulation for bootstrapping a POMDP dialogue system. In *Proceedings of HLT/NAACL 2007*, pages 149–152, Rochester, NY.
Schatzmann, J., K. Weilhammer, M. N. Stuttle, and S. J. Young. 2006. A survey of statistical user simulation techniques for reinforcement learning of dialogue management strategies. *Knowledge Engineering Review*, 21:97–126.
Shapiro, D. and P. Langley. 2002. Separating skills from preference: Using learning to program by reward. In *Proceedings ICML-02*, pages 570–577, Sydney.
Strauss, P. M., H. Hoffmann, and S. Scherer. 2007. Evaluation and user acceptance of a dialogue system using Wizard-of-Oz recordings. In *Proceedings of 3rd IET International Conference on Intelligent Environments*, pages 521–524, Germany.
Sutton, R. and A. Barto. 1998. *Reinforcement Learning*. MIT Press, Cambridge, MA.
Vygotsky, L. S. 1978. *Mind in Society: The Development of Higher Psychological Processes*. Harvard University Press, Cambridge, MA.
Whittaker, S., M. Walker, and J. Moore. 2002. Fish or fowl: A Wizard of Oz evaluation of dialogue strategies in the restaurant domain. In *Language Resources and Evaluation Conference*, pages 1,602–1,609, Las Palmas.
Williams, J. and S. J. Young. 2003. Using Wizard-of-Oz simulations to bootstrap reinforcement learning based dialogue management systems. In *Proceedings SIGdial'03*, pages 135–139, Sapporo.