Learning to Adapt to Unknown Users: Referring Expression Generation in Spoken Dialogue Systems

Srinivasan Janarthanam
School of Informatics
University of Edinburgh
`s.janarthanam@ed.ac.uk`

Oliver Lemon
Interaction Lab
Mathematics and Computer Science (MACS)
Heriot-Watt University
`o.lemon@hw.ac.uk`

Abstract

We present a data-driven approach to learn user-adaptive referring expression generation (REG) policies for spoken dialogue systems. Referring expressions can be difficult to understand in technical domains where users may not know the technical 'jargon' names of the domain entities. In such cases, dialogue systems must be able to model the user's (lexical) domain knowledge and use appropriate referring expressions. We present a reinforcement learning (RL) framework in which the system learns REG policies which can adapt to unknown users online. Furthermore, unlike supervised learning methods which require a large corpus of expert adaptive behaviour to train on, we show that effective adaptive policies can be learned from a small dialogue corpus of non-adaptive human-machine interaction, by using a RL framework and a statistical user simulation. We show that in comparison to adaptive hand-coded baseline policies, the learned policy performs significantly better, with an 18.6% average increase in adaptation accuracy. The best learned policy also takes less dialogue time (average 1.07 min less) than the best hand-coded policy. This is because the learned policies can adapt online to changing evidence about the user's domain expertise.

1 Introduction

We present a reinforcement learning (Sutton and Barto, 1998) framework to learn user-adaptive referring expression generation policies from data-driven user simulations. A user-adaptive REG policy allows the system to choose appropriate expressions to refer to domain entities in a dialogue setting. For instance, in a technical support conversation, the system could choose to use more technical terms with an expert user, or to use more descriptive and general expressions with novice users, and a mix of the two with intermediate users of various sorts (see examples in Table 1).

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

Table 1: Referring expression examples for 2 entities (from the corpus)

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). We assume that users are mostly unknown to the system and therefore that a spoken dialogue system (SDS) must be capable of observing the user's dialogue behaviour, modelling his/her domain knowledge, and adapting accordingly, just like human interlocutors. Rule-based and supervised learning approaches to user adaptation in SDS have been proposed earlier (Cawsey, 1993; Akiba and Tanaka, 1994). However, such methods require expensive resources such as domain experts to hand-code the rules, or a corpus of expert-layperson interactions to train on. In contrast, we present a corpus-driven framework using which a user-adaptive REG policy can be learned using RL from a small corpus of non-adaptive human-machine interaction.

We show that these learned policies perform better than simple hand-coded adaptive policies in terms of accuracy of adaptation and dialogue time. We also compared the performance of policies learned using a hand-coded rule-based simulation and a data-driven statistical simulation and show that data-driven simulations produce better policies than rule-based ones.

In section 2, we present some of the related work. Section 3 presents the dialogue data that we used to train the user simulation. Section 4 and section 5 describe the dialogue system framework and the user simulation models. In section 6, we present the training and in section 7, we present the evaluation for different REG policies.

2 Related work

There are several ways in which natural language generation (NLG) systems adapt to users. Some of them adapt to a user's goals, preferences, environment and so on. Our focus in this study is restricted to the user's lexical domain expertise. Several NLG systems adapt to the user's domain expertise at different levels of generation - text planning (Paris, 1987), complexity of instructions (Dale, 1989), referring expressions (Reiter, 1991), and so on. Some dialogue systems, such as COMET, have also incorporated NLG modules that present appropriate levels of instruction to the user (McKeown et al., 1993). However, in all the above systems, the user's knowledge is assumed to be accurately represented in an initial user model using which the system adapts its language. In contrast to all these systems, our adaptive REG policy knows nothing about the user when the conversation starts.

Rule-based and supervised learning approaches have been proposed to learn and adapt during the conversation dynamically. Such systems learned from the user at the start and later adapted to the domain knowledge of the users. However, they either require expensive expert knowledge resources to hand-code the inference rules (Cawsey, 1993) or large corpus of expert-layperson interaction from which adaptive strategies can be learned and modelled, using methods such as Bayesian networks (Akiba and Tanaka, 1994). In contrast, we present an approach that learns in the absence of these expensive resources. It is also not clear how supervised and rule-based approaches choose between when to seek more information and when to adapt. In this study, we show that using reinforcement learning this decision is learned automatically.

Reinforcement Learning (RL) has been successfully used for learning dialogue management policies since (Levin et al., 1997). The learned policies allow the dialogue manager to optimally choose appropriate dialogue acts such as instructions, confirmation requests, and so on, under uncertain noise or other environment conditions. There have been recent efforts to learn information presentation and recommendation strategies using reinforcement learning (Rieser and Lemon, 2009; Hernandez et al., 2003; Rieser and Lemon, 2010), and joint optimisation of Dialogue Management and NLG using hierarchical RL has been proposed by (Lemon, 2010). In contrast, we present a framework to learn to choose appropriate referring expressions based on a user's domain knowledge. Earlier, we reported a proof-of-concept work using a hand-coded rule-based user simulation (Janarthanam and Lemon, 2009c).

3 The Wizard-of-Oz Corpus

We use a corpus of technical support dialogues collected from real human users using a Wizard-of-Oz method (Janarthanam and Lemon, 2009b). The corpus consists of 17 dialogues from users who were instructed to physically set up a home broadband connection using objects like a wireless modem, cables, filters, etc. They listened to the instructions from the system and carried them out using the domain objects laid in front of them. The human 'wizard' played the role of only an interpreter who would understand what the user said and annotate it as a dialogue act. The set-up examined the effect of using three types of referring expressions (jargon, descriptive, and tutorial), on the users.

Out of the 17 dialogues, 6 used a jargon strategy, 6 used a descriptive strategy, and 5 used a tutorial strategy[1]. The task had reference to 13 domain entities, mentioned repeatedly in the dialogue. In total, there are 203 jargon, 202 descriptive and 167 tutorial referring expressions. Interestingly, users who weren't acquainted with the domain objects requested clarification on some of the referring expressions used. The dialogue exchanges between the user and system were logged in the form of dialogue acts and the system's choices of referring expressions. Each user's knowledge of domain entities was recorded both before and after the task and each user's interac-

[1]The tutorial strategy uses both jargon and descriptive expressions together.

tions with the environment were recorded. We use the dialogue data, pre-task knowledge tests, and the environment interaction data to train a user simulation model. Pre and post-task test scores were used to model the learning behaviour of the users during the task (see section 5).

The corpus also recorded the time taken to complete each dialogue task. We used these data to build a regression model to calculate total dialogue time for dialogue simulations. The strategies were never mixed (with some jargon, some descriptive and some tutorial expressions) within a single conversation. Therefore, please note that the strategies used for data collection were *not adaptive* and the human 'wizard' has no role in choosing which referring expression to present to the user. Due to this fact, no user score regarding adaptation was collected. We therefore measure adaptation objectively as explained in section 6.1.

4 The Dialogue System

In this section, we describe the different modules of the dialogue system. The interaction between the different modules is shown in figure 1 (in learning mode). The dialogue system presents the user with instructions to setup a broadband connection at home. In the Wizard of Oz setup, the system and the user interact using speech. However, in our machine learning setup, they interact at the abstract level of dialogue actions and referring expressions. Our objective is to learn to choose the appropriate referring expressions to refer to the domain entities in the instructions.

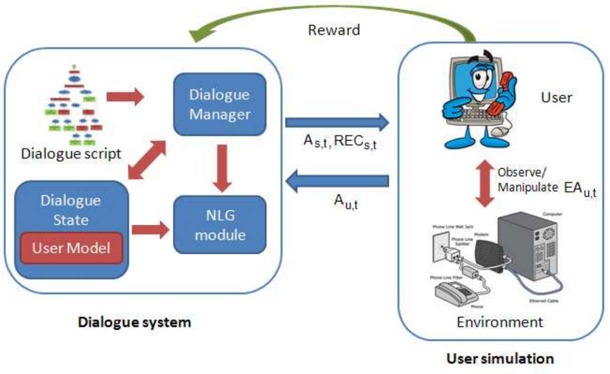

Figure 1: System User Interaction (learning)

4.1 Dialogue Manager

The dialogue manager identifies the next instruction (dialogue act) to give to the user based on the dialogue management policy π_{dm}. Since, in this study, we focus only on learning the REG policy, the dialogue management is coded in the form of a finite state machine. In this dialogue task, the system provides two kinds of instructions - observation and manipulation. For observation instructions, users observe the environment and report back to the system, and for the manipulation instructions (such as plugging in a cable in to a socket), they manipulate the domain entities in the environment. When the user carries out an instruction, the system state is updated and the next instruction is given. Sometimes, users do not understand the referring expressions used by the system and then ask for clarification. In such cases, the system provides clarification on the referring expression ($provide\_clar$), which is information to enable the user to associate the expression with the intended referent. The system action $A_{s,t}$ (t denoting turn, s denoting system) is therefore to either give the user the next instruction or a clarification. When the user responds in any other way, the instruction is simply repeated. The dialogue manager is also responsible for updating and managing the system state $S_{s,t}$ (see section 4.2). The system interacts with the user by passing both the system action $A_{s,t}$ and the referring expressions $REC_{s,t}$ (see section 4.3).

4.2 The dialogue state

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

Each jargon referring expression x is represented by a three valued variable in the dialogue state: user_knows_x. The three values that each variable takes are yes, no, not_sure. The variables are updated using a simple user model update algorithm. Initially each variable is set to not_sure. If the user responds to an instruction containing the referring expression x with a clarification request, then user_knows_x is set to no. Similarly, if the user responds with appropriate information to the system's instruction, the dialogue manager sets user_knows_x is set to yes.

The dialogue manager updates the variables concerning the referring expressions used in the current system utterance appropriately after the user's response each turn. The user may have the capacity to learn jargon. However, only the user's initial knowledge is recorded. This is based on the assumption that an estimate of the user's knowledge helps to predict the user's knowledge of the rest of the referring expressions. Another issue concerning the state space is its size. Since, there are 13 entities and we only model the jargon expressions, the state space size is 3^{13}.

4.3 REG module

The REG module is a part of the NLG module whose task is to identify the list of domain entities to be referred to and to choose the appropriate referring expression for each of the domain entities for each given dialogue act. In this study, we focus only on the production of appropriate referring expressions to refer to domain entities mentioned in the dialogue act. It chooses between the two types of referring expressions - jargon and descriptive. For example, the domain entity *broadband_filter* can be referred to using the jargon expression "broadband filter" or using the descriptive expression "small white box"[2]. We call this the act of choosing the *REG action*. The tutorial strategy was not investigated here since the corpus analysis showed tutorial utterances to be very time consuming. In addition, they do not contribute to the adaptive behaviour of the system.

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

$$REC_{s,t} = \{(R_1, T_1), ..., (R_n, T_n)\}$$

In the evaluation mode, a trained REG policy interacts with unknown users. It consults the learned policy π_{reg} to choose the referring expressions based on the current user model.

5 User Simulations

In this section, we present user simulation models that simulate the dialogue behaviour of a real human user. These external simulation models are different from internal user models used by the dialogue system. In particular, our model is the first to be sensitive to a system's choices of referring expressions. The simulation has a statistical distribution of in-built knowledge profiles that determines the dialogue behaviour of the user being simulated. If the user does not know a referring expression, then he is more *likely* to request clarification. If the user is able to interpret the referring expressions and identify the references then he is more likely to follow the system's instruction. This behaviour is simulated by the action selection models described below.

Several user simulation models have been proposed for use in reinforcement learning of dialogue policies (Georgila et al., 2005; Schatzmann et al., 2006; Schatzmann et al., 2007; Ai and Litman, 2007). However, they are suited only for learning dialogue management policies, and not natural language generation policies. Earlier, we presented a two-tier simulation trained on data precisely for REG policy learning (Janarthanam and Lemon, 2009a). However, it is not suited for training on small corpus like the one we have at our disposal. In contrast to the earlier model, we now condition the clarification requests on the referent class rather than the referent itself to handle data sparsity problem.

The user simulation (US) receives the system action $A_{s,t}$ and its referring expression choices $REC_{s,t}$ at each turn. The US responds with a user action $A_{u,t}$ (u denoting user). This can either be a clarification request (cr) or an instruction

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

response (ir). We used two kinds of action selection models: corpus-driven statistical model and hand-coded rule-based model.

5.1 Corpus-driven action selection model

In the corpus-driven model, the US produces a clarification request cr based on the class of the referent $C(R_i)$, type of the referring expression T_i, and the current domain knowledge of the user for the referring expression $DK_{u,t}(R_i, T_i)$. Domain entities whose jargon expressions raised clarification requests in the corpus were listed and those that had more than the mean number of clarification requests were classified as `difficult` and others as `easy` entities (for example, "power adaptor" is `easy` - all users understood this expression, "broadband filter" is `difficult`). Clarification requests are produced using the following model.

$$P(A_{u,t} = cr(R_i, T_i) | C(R_i), T_i, DK_{u,t}(R_i, T_i))$$
$$\text{where } (R_i, T_i) \in REC_{s,t}$$

One should note that the actual literal expression is not used in the transaction. Only the entity that it is referring to (R_i) and its type (T_i) are used. However, the above model simulates the process of interpreting and resolving the expression and identifying the domain entity of interest in the instruction. The user identification of the entity is signified when there is no clarification request produced (i.e. $A_{u,t} = none$). When no clarification request is produced, the environment action $EA_{u,t}$ is generated using the following model.

$$P(EA_{u,t}|A_{s,t}) \text{ if } A_{u,t}! = cr(R_i, T_i)$$

Finally, the user action is an instruction response which is determined by the system action $A_{s,t}$. Instruction responses can be different in different conditions. For an observe and report instruction, the user issues a $provide\_info$ action and for a manipulation instruction, the user responds with an $acknowledgement$ action and so on.

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

All the above models were trained on our corpus data using *maximum likelihood estimation* and smoothed using a variant of *Witten-Bell discounting*. According to the data, clarification requests are much more likely when jargon expressions are used to refer to the referents that belong to the `difficult` class and which the user doesn't know about. When the system uses expressions that the user knows, the user generally responds to the instruction given by the system. These user simulation models have been evaluated and found to produce behaviour that is very similar to the original corpus data, using the Kullback-Leibler divergence metric (Cuayahuitl, 2009).

livebox $= 1$	*power_adaptor* $= 1$
wall_phone_socket $= 1$	*broadband_filter* $= 0$
broadband_cable $= 0$	*ethernet_cable* $= 1$
lb_power_light $= 1$	*lb_power_socket* $= 1$
lb_broadband_light $= 0$	*lb_ethernet_light* $= 0$
lb_adsl_socket $= 0$	*lb_ethernet_socket* $= 0$
pc_ethernet_socket $= 1$	

Table 2: Domain knowledge: an Intermediate User

5.2 Rule-based action selection model

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

5.3 User Domain knowledge

The user domain knowledge is initially set to one of several models at the start of every conversation. The models range from novices to experts which were identified from the corpus using `k-means` clustering. The initial knowledge base ($DK_{u,initial}$) for an intermediate user is shown in table 2. A novice user knows only "power adaptor", and an expert knows all the jargon expressions. We assume that users can interpret the descriptive expressions and resolve their references. Therefore, they are not explicitly represented. We only code the user's knowledge of jargon expressions. This is represented by a boolean variable for each domain entity.

Corpus data shows that users can learn jargon expressions during the conversation. The user's domain knowledge DK_u is modelled to be dynamic and is updated during the conversation. Based on our data, we found that when presented with clarification on a jargon expression, users always learned the jargon.

$$\begin{gathered}\text{if } A_{s,t} = provide\_clar(R_i, T_i) \\ DK_{u,t+1}(R_i, T_i) \leftarrow 1\end{gathered}$$

Users also learn when jargon expressions are repeatedly presented to them. Learning by repetition follows the pattern of a learning curve - the greater the number of repetitions $\#(R_i, T_i)$, the higher the likelihood of learning. This is modelled stochastically based on repetition using the parameter $\#(R_i, T_i)$ as follows (where $(R_i, T_i) \in REC_{s,t}$) .

$$P(DK_{u,t+1}(R_i, T_i) \leftarrow 1 | \#(R_i, T_i))$$

The final state of the user's domain knowledge ($DK_{u,final}$) may therefore be different from the initial state ($DK_{u,initial}$) due to the learning effect produced by the system's use of jargon expressions. In most studies done previously, the user's domain knowledge is considered to be static. However in real conversation, we found that the users nearly always learned jargon expressions from the system's utterances and clarifications.

6 Training

The REG module was trained (operated in learning mode) using the above simulations to learn REG policies that select referring expressions based on the user expertise in the domain. As shown in figure 1, the learning agent (REG module) is given a reward at the end of every dialogue. During the training session, the learning agent explores different ways to maximize the reward. In this section, we discuss how to code the learning agent's goals as reward. We then discuss how the reward function is used to train the learning agent.

6.1 Reward function

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

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

$$\begin{gathered}RA_r = \frac{\#(appropriate\_expressions(r))}{\#(instances(r))} \\ AdaptationAccuracy AA = \frac{1}{\#(r)} \Sigma_r RA_r\end{gathered}$$

Note that this reward is computed at the end of the dialogue (it is a 'final' reward), and is then back-propagated along the action sequence that led to that final state. Thus the reward can be computed for each system REG action, without the system having access to the user's initial domain knowledge while it is learning a policy.

Since the agent starts the conversation with no knowledge about the user, it may try to use more exploratory moves to learn about the user, although they may be inappropriate. However, by measuring accuracy to the initial user state, the agent is encouraged to restrict its exploratory moves and start predicting the user's domain knowledge as soon as possible. The system should therefore ideally explore less and adapt more to increase accuracy. The above reward function returns 1 when the agent is completely accurate in adapting to the user's domain knowledge and it returns 0 if the agent's REC choices were completely inappropriate. Usually during learning, the reward value lies between these two extremes and the agent tries to maximize it to 1.

6.2 Learning

The REG module was trained in learning mode using the above reward function using the SHARSHA reinforcement learning algorithm (with linear function approximation) (Shapiro and Langley, 2002). This is a hierarchical variant of SARSA, which is an on-policy learning algorithm that updates the current behaviour policy (see (Sutton and Barto, 1998)). The training produced approx. 5000 dialogues. Two types of simulations were used as described above: Data-driven and Hand-coded. Both user simulations were calibrated to produce three types of users: Novice, Int2 (intermediate) and Expert, randomly but with equal probability. Novice users knew just one jargon expression, Int2 knew seven, and Expert users knew all thirteen jargon expressions. There was an underlying pattern in these knowledge profiles. For example, Intermediate users were those who knew the commonplace domain entities but not those specific to broadband connection. For instance, they knew "ethernet cable" and "pc ethernet socket" but not "broadband filter" and "broadband cable".

Initially, the REG policy chooses randomly between the referring expression types for each domain entity in the system utterance, irrespective of the user model state. Once the referring expressions are chosen, the system presents the user simulation with both the dialogue act and referring expression choices. The choice of referring expression affects the user's dialogue behaviour which in turn makes the dialogue manager update the user model. For instance, choosing a jargon expression could evoke a clarification request from the user, which in turn prompts the dialogue manager to update the user model with the new information that the user is ignorant of the particular expression. It should be noted that using a jargon expression is an *information seeking* move which enables the REG module to estimate the user's knowledge level. The same process is repeated for every dialogue instruction. At the end of the dialogue, the system is rewarded based on its choices of referring expressions. If the system chooses jargon expressions for novice users or descriptive expressions for expert users, penalties are incurred and if the system chooses REs appropriately, the reward is high. On the one hand, those actions that fetch more reward are reinforced, and on the other hand, the agent tries out new state-action combinations to explore the possibility of greater rewards. Over time, it stops exploring new state-action combinations and exploits those actions that contribute to higher reward. The REG module learns to choose the appropriate referring expressions based on the user model in order to maximize the overall adaptation accuracy.

Figure 2 shows how the agent learns using the data-driven (**Learned DS**) and hand-coded simulations (**Learned HS**) during training. It can be seen in the figure 2 that towards the end the curve plateaus signifying that learning has converged.

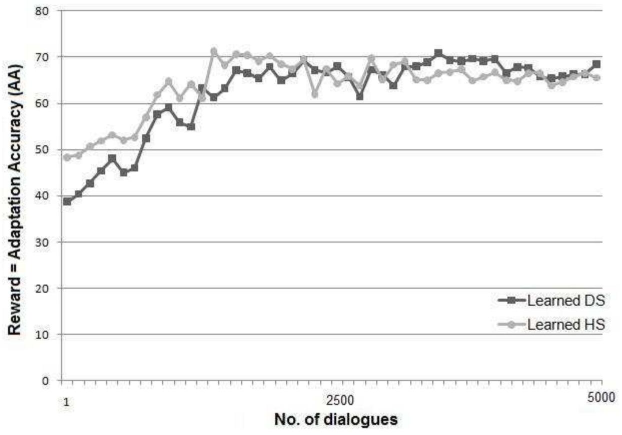

Figure 2: Learning curves - Training

7 Evaluation

In this section, we present the evaluation metrics used, the baseline policies that were hand-coded for comparison, and the results of evaluation.

7.1 Metrics

In addition to the adaptation accuracy mentioned in section 6.1, we also measure other parameters from the conversation in order to show how learned adaptive policies compare with other policies on other dimensions. We calculate the time taken ($Time$) for the user to complete the dialogue task. This is calculated using a regression model from the corpus based on number of words, turns, and mean user response time. We also measure the (normalised) learning gain (LG) produced by using unknown jargon expressions. This is calculated using the pre and post scores from the user domain knowledge (DK_u) as follows.

$$Learning\ Gain\ LG = \frac{Post-Pre}{1-Pre}$$

7.2 Baseline REG policies

In order to compare the performance of the learned policy with hand-coded REG policies, three simple rule-based policies were built. These were built in the absence of expert domain knowledge and a expert-layperson corpus.

- Jargon: Uses jargon for all referents by default. Provides clarifications when requested.
- Descriptive: Uses descriptive expressions for all referents by default.
- Switching: This policy starts with jargon expressions and continues using them until the user requests for clarification. It then switches to descriptive expressions and continues to use them until the user complains. In short, it switches between the two strategies based on the user's responses.

All the policies exploit the user model in subsequent references after the user's knowledge of the expression has been set to either `yes` or `no`. Therefore, although these policies are simple, they do adapt to a certain extent, and are reasonable baselines for comparison in the absence of expert knowledge for building more sophisticated baselines.

7.3 Results

The policies were run under a testing condition (where there is no policy learning or exploration) using a data-driven simulation calibrated to simulate 5 different user types. In addition to the three users - Novice, Expert and Int2, from the training simulations, two other intermediate users (Int1 and Int3) were added to examine how well each policy handles unseen user types. The REG module was operated in evaluation mode to produce around 200 dialogues per policy distributed over the 5 user groups.

Overall performance of the different policies in terms of $Adaptation\ Accuracy\ (AA)$, $Time$ and $Learning\ Gain\ (LG)$ are given in Table 3. Figure 3 shows how each policy performs in terms of accuracy on the 5 types of users.

We found that the Learned DS policy (i.e. learned with the data-driven user simulation) is the most accurate (Mean = 79.70, SD = 10.46) in terms of adaptation to each user's initial state of domain knowledge. Also, it is the only policy that has more or less the same accuracy scores

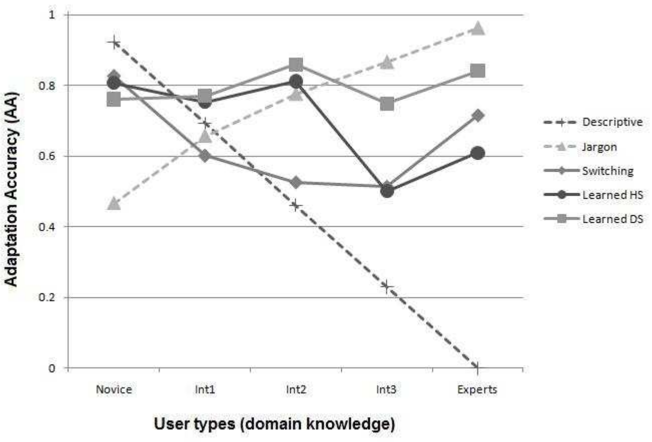

Figure 3: Evaluation - Adaptation Accuracy

Policies	AA	Time T	LG
Descriptive	46.15	7.44	0
Jargon	74.54	9.15*	0.97*
Switching	62.47	7.48	0.30
Learned HS	69.67	7.52	0.33
Learned DS	**79.70***	8.08*	0.63*

* Significantly different from all others ($p < 0.05$).

Table 3: Evaluation on 5 user types

over all different user types (see figure 3). It should also be noted that the it generalised well over user types (Int1 and Int3) which were unseen in training. Learned DS policy outperforms all other policies: Learned HS (Mean = 69.67, SD = 14.18), Switching (Mean = 62.47, SD = 14.18), Jargon (Mean = 74.54, SD = 17.9) and Descriptive (Mean = 46.15, SD = 33.29). The differences between the accuracy (AA) of the Learned DS policy and all other policies were statistically significant with $p < 0.05$ (using a two-tailed paired t-test). Although Learned HS policy is similar to the Learned DS policy, as shown in the learning curves in figure 2, it does not perform as well when confronted with users types that it did not encounter during training. The Switching policy, on the other hand, quickly switches its strategy (sometimes erroneously) based on the user's clarification requests but does not adapt appropriately to evidence presented later during the conversation. Sometimes, this policy switches erroneously because of the uncertain user behaviours. In contrast, learned policies continuously adapt to new evidence. The Jargon policy performs better than the Learned HS and Switching policies. This because the system can learn more about the user by using more jargon expressions and then use that knowledge for adaptation for known referents. However, it is not possible for this policy to predict the user's knowledge of unseen referents. The Learned DS policy performs better than the Jargon policy, because it is able to accurately predict the user's knowledge of referents unseen in the dialogue so far.

The learned policies are a little more time-consuming than the Switching and Descriptive policies but compared to the Jargon policy, Learned DS takes 1.07 minutes less time. This is because learned policies use a few jargon expressions (giving rise to clarification requests) to learn about the user. On the other hand, the Jargon policy produces more user learning gain because of the use of more jargon expressions. Learned policies compensate on time and learning gain in order to predict and adapt well to the users' knowledge patterns. This is because the training was optimized for accuracy of adaptation and not for learning gain or time taken. The results show that using our RL framework, REG policies can be learned using data-driven simulations, and that such a policy can predict and adapt to a user's knowledge pattern more accurately than policies trained using hand-coded rule-based simulations and hand-coded baseline policies.

7.4 Discussion

The learned policies explore the user's expertise and predict their knowledge patterns, in order to better choose expressions for referents unseen in the dialogue so far. The system learns to identify the patterns of knowledge in the users with a little exploration (information seeking moves). So, when it is provided with a piece of evidence (e.g. the user knows "broadband filter"), it is able to accurately estimate unknown facts (e.g. the user might know "broadband cable"). Sometimes, its choices are wrong due to incorrect estimation of the user's expertise (due to stochastic behaviour of the users). In such cases, the incorrect adaptation move can be considered to be an information seeking move. This helps further adaptation using the new evidence. By continuously using this "seek-predict-adapt" approach, the system adapts dynamically to different users. Therefore, with a little information seeking and better prediction, the learned policies are able to better adapt to users with different domain expertise.

In addition to adaptation, learned policies learn to identify when to seek information from the user to populate the user model (which is initially set to `not_sure`). It should be noted that the system cannot adapt unless it has some information about the user and therefore needs to decisively seek information by using jargon expressions. If it seeks information all the time, it is not adapting to the user. The learned policies therefore learn to trade-off between information seeking moves and adaptive moves in order to maximize the overall adaptation accuracy score.

8 Conclusion

In this study, we have shown that user-adaptive REG policies can be learned from a small corpus of non-adaptive dialogues between a dialogue system and users with different domain knowledge levels. We have shown that such adaptive REG policies learned using a RL framework adapt to unknown users better than simple hand-coded policies built without much input from domain experts or from a corpus of expert-layperson adaptive dialogues. The learned, adaptive REG policies learn to trade off between adaptive moves and information seeking moves automatically to maximize the overall adaptation accuracy. Learned policies start the conversation with information seeking moves, learn a little about the user, and start adapting dynamically as the conversation progresses. We have also shown that a data-driven statistical user simulation produces better policies than a simple hand-coded rule-based simulation, and that the learned policies generalise well to unseen users.

In future work, we will evaluate the learned policies with real users to examine how well they adapt, and examine how real users evaluate them (subjectively) in comparison to baselines. Whether the learned policies perform better or as well as a hand-coded policy 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 behaviour and adapt accordingly.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 216594 (CLASSiC project `www.classic-project.org`) and from the EPSRC, project no. EP/G069840/1.

References

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

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

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

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

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

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

R. Dale. 1989. Cooking up referring expressions. In *Proc. ACL-1989.*

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

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

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

S. Janarthanam and O. Lemon. 2009a. A Two-tier User Simulation Model for Reinforcement Learning of Adaptive Referring Expression Generation Policies. In *Proc. SigDial'09.*

S. Janarthanam and O. Lemon. 2009b. A Wizard-of-Oz environment to study Referring Expression Generation in a Situated Spoken Dialogue Task. In *Proc. ENLG'09.*

S. Janarthanam and O. Lemon. 2009c. Learning Lexical Alignment Policies for Generating Referring Expressions for Spoken Dialogue Systems. In *Proc. ENLG'09.*

O. Lemon. 2010. Learning what to say and how to say it: joint optimization of spoken dialogue management and Natural Language Generation. *Computer Speech and Language.* (to appear).

E. Levin, R. Pieraccini, and W. Eckert. 1997. Learning Dialogue Strategies within the Markov Decision Process Framework. In *Proc. of ASRU97.*

K. McKeown, J. Robin, and M. Tanenblatt. 1993. Tailoring Lexical Choice to the User's Vocabulary in Multimedia Explanation Generation. In *Proc. ACL 1993.*

C. L. Paris. 1987. *The Use of Explicit User Models in Text Generations: Tailoring to a User's Level of Expertise*. Ph.D. thesis, Columbia University.

E. Reiter. 1991. Generating Descriptions that Exploit a User's Domain Knowledge. In R. Dale, C. Mellish, and M. Zock, editors, *Current Research in Natural Language Generation*, pages 257–285. Academic Press.

V. Rieser and O. Lemon. 2009. Natural Language Generation as Planning Under Uncertainty for Spoken Dialogue Systems. In *Proc. EACL'09.*

V. Rieser and O. Lemon. 2010. Optimising information presentation for spoken dialogue systems. In *Proc. ACL.* (to appear).

J. Schatzmann, 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*, pages 97–126.

J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. J. Young. 2007. Agenda-based User Simulation for Bootstrapping a POMDP Dialogue System. In *Proc of HLT/NAACL 2007.*

D. Shapiro and P. Langley. 2002. Separating skills from preference: Using learning to program by reward. In *Proc. ICML-02.*

R. Sutton and A. Barto. 1998. *Reinforcement Learning*. MIT Press.