D4.5: Testing learned NLG and TTS policies with real users, in Self-Help and Appointment Scheduling Systems

Srini Janarthanam, Oliver Lemon, Romain Laroche, and Ghislain Putois

Distribution: Public
The partners in CLASSiC are:

- Heriot-Watt University (HWU)
- University of Cambridge (UCAM)
- University of Geneva (GENE)
- Ecole Superieure d'Electricite (SUPELEC)
- France Telecom/Orange Labs (FT)
- University of Edinburgh (HCRC or EDIN)

For copies of reports, updates on project activities and other CLASSiC-related information, contact:

The CLASSiC Project Co-ordinator:
Dr. Oliver Lemon
School of Mathematical and Computer Sciences (MACS)
Heriot-Watt University
Edinburgh
EH14 4AS
United Kingdom
O.Lemon@hw.ac.uk
Phone +44 (131) 451 3782 - Fax +44 (0)131 451 3327

Copies of reports and other material can also be accessed via the project’s administration homepage, http://www.classic-project.org

©2011, The Individual Authors.
Contents

Executive Summary 1

1. Testing adaptive NLG policies with real users 2
   1.1. Related work 2
   1.2. The Dialogue System 3
      1.2.1. Dialogue Manager 3
      1.2.2. The dialogue state 4
      1.2.3. REG module 4
   1.3. User Simulations 5
      1.3.1. Corpus-driven action selection model 5
      1.3.2. User Domain knowledge 6
      1.3.3. Corpus 6
   1.4. Training 6
      1.4.1. Reward function 6
      1.4.2. Learning 7
   1.5. Evaluation 7
      1.5.1. Baseline system 7
      1.5.2. Experimental setup 8
      1.5.3. Metrics 9
      1.5.4. Results 9
      1.5.5. Analysis 10
   1.6. Conclusion 11

2. Testing TTS policies with real users 12
   2.1. Introduction 12
   2.2. Raw performances 12
   2.3. Task completion rate on nominal conditions 13
   2.4. Internal and effective task completion rates 13
   2.5. Perceived task completion rate 13
   2.6. Conclusion for TTS in CLASSiC System 3 14

3. Summary and Conclusion 16

A. Appendix A: Evaluation of trained NLG in TownInfo (System 1) 19

B. Appendix B: Evaluation of trained NLG in Appointment Scheduling System (System 2) 19
Executive summary

We present results from two different real-user evaluations of data-driven approaches to Natural Language Generation (NLG) and text-to-speech (TTS). The first concerns user-adaptive referring expression generation (REG) in a Self-Help system, and the second concerns TTS variants (the use of different voices) for an Appointment Scheduling (AS) Spoken Dialogue System.

The first evaluation focuses on the problem that 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 developed a reinforcement learning (RL) framework in which the system learns REG policies which can adapt to unknown users online (published at ACL 2010: [Janarthanam and Lemon, 2010b]). For real users of such a system, we show that in comparison to an adaptive hand-coded baseline policy, the learned policy performs significantly better, with a 20.8% increase in adaptation accuracy, 11% decrease in time taken, and a 14.7% increase in task completion rate. An initial version of his work was published at SIGDIAL 2010 [Janarthanam and Lemon, 2010a]. The learned policy also has a significantly better subjective rating from users. This is because the learned policies adapt online to changing evidence about the user’s domain expertise. We also discuss the issue of evaluation in simulation versus evaluation with real users.

Note that a reinforcement learning approach to adaptive Temporal Expression Generation (to appear at SIGDIAL 2011: [Janarthanam et al., 2011]) was also evaluated in the context of the overall evaluation of the CLASSiC “System 2” Appointment Scheduling system. This evaluation is therefore reported in deliverable 6.4. In addition, we also evaluated a trained NLG component for Information Presentation (recommendation) in the TownInfo system (“System 1”). This evaluation is also reported in D6.4, in the context of full system evaluations. (Please see the Appendices for brief summaries of these evaluations.)

The second evaluation reported here tests an adaptive policy for selecting amongst voices for carefully handcrafted TTS prompts. The evaluation was performed on CLASSiC System 3, which aims at improving an Appointment Scheduling task. No significant improvement is observed in this task. We discuss the plausible causes of such a result.
1. Testing adaptive NLG policies with real users

We present results from an evaluation with real users, for a reinforcement learning [Sutton and Barto, 1998] framework to learn user-adaptive referring expression generation policies from data-driven user simulations for a Self-Help spoken dialogue system [Janarthanam and Lemon, 2010b]. Such a 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. Therefore unlike systems that use static user models, our system has to dynamically model the user’s domain knowledge in order to adapt during the conversation.

We present a corpus-driven framework for learning a user-adaptive REG policy from a small corpus of non-adaptive human-machine interaction. We show that the learned policy performs better than a simple hand-coded adaptive policy in terms of accuracy of adaptation, dialogue time and task completion rate when evaluated with real users in a wizarded study.

In Section 1.1, we present some of the related work. Section 1.2 and Section 1.3 describe the dialogue system framework and the user simulation model. In Section 1.4, we present the training and in Section 1.5, we present the evaluation for different REG policies with real users.

1.1. Related work

Rule-based and supervised learning approaches have been proposed to learn and adapt during conversations dynamically. Such systems learn from a user at the start and later adapt to the domain knowledge of the user. However, they either require expensive expert knowledge resources to hand-code the inference rules [Cawsey, 1993] or a 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 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 [Hernandez et al., 2003, Rieser and Lemon, 2009, Rieser and Lemon, 2010], and joint optimisation of Dialogue Management and NLG using hierarchical RL has been proposed by [Lemon, 2011]. In addition, we present a framework to learn to choose appropriate referring expressions based on a user’s domain knowledge. Following a proof-of-concept study using a hand-coded rule-based user simulation [Janarthanam and Lemon, 2009c], we previously showed that adaptive REG policies can be learned using an RL framework with data-driven user simulations and that such policies perform better than simple hand-coded policies [Janarthanam and Lemon, 2010b].

1.2. 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)](image)

1.2.1. Dialogue Manager

The dialogue manager identifies the next dialogue act ($A_{st}$ where $t$ denotes turn, $s$ denotes system) to give to the user based on the dialogue management policy $\pi_{dm}$. The dialogue management is coded in the form of a finite state machine. In this dialogue task, the system provides instructions to either
observe or manipulate the environment. When users ask for clarifications on referring expressions, the system clarifies (provide clar) by giving information to enable the user to associate the expression with the intended referent. 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 1.2.2). The system interacts with the user by passing both the system action $A_s, t$ and the referring expressions $REC_s, t$ (see Section 1.2.3).

### 1.2.2. 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 knowledge about the user. Since the model is updated according to the user’s behaviour, it may be inaccurate if the user’s behaviour is itself uncertain. Hence, 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: $\text{user_knows}_x$. The three values that each variable takes are yes, no, not sure. The variables are updated using a simple user model update algorithm after the user’s response each turn. 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 $\text{user_knows}_x$ is set to no. Similarly, if the user responds with appropriate information to the system’s instruction, the dialogue manager sets $\text{user_knows}_x$ is set to yes. Only the user’s initial knowledge is recorded. This is based on the assumption that an estimate of the user’s initial knowledge helps to predict the user’s knowledge of the rest of the referring expressions.

### 1.2.3. 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”¹. Although adaptation is the primary goal, it should be noted that in order to get an idea of the user the system is dealing with, it needs to seek information using jargon expressions.

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 referring expressions. This is represented by a REG policy $\pi_{\text{reg}} : UMs, t \rightarrow RECs, t$, which maps the states of the dialogue (user model) to optimal referring expressions. The referring expression choices $RECs, t$ 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”.

$$RECs, t = \{(R_1, T_1), \ldots, (R_n, T_n)\}$$

¹We 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”).
In the evaluation mode, a trained REG policy interacts with unknown users. It consults the learned policy $\pi_{reg}$ to choose the referring expressions based on the current user model.

### 1.3. User Simulations

In this section, we present user simulation models that simulate the dialogue behaviour of a real human user. 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. In particular, our model is the first to be sensitive to a system’s choices of referring expressions. 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 the data sparsity problem.

#### 1.3.1. Corpus-driven action selection model

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 response ($ir$). 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.

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

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.

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

Finally, the user action is an instruction response which is determined by the system action $A_{s,t}$. Instruction responses can be either *provide_info*, *acknowledgement* or *other* based on the system’s instruction.

$$P(A_{u,t} = ir|EA_{u,t}, A_{s,t})$$
All the above models were trained on our corpus data using maximum likelihood estimation and smoothed using a variant of Witten-Bell discounting. The corpus contained dialogues between a non-adaptive dialogue system and real users. 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.

1.3.2. 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. A novice user knows only “power adaptor”, an expert knows all the jargon expressions and intermediate users know some. 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 using boolean variables representing whether the user knows the expression or not.

1.3.3. Corpus

We trained the action selection model on a small corpus of 12 non-adaptive dialogues between real users and a dialogue system. There were six dialogues in which users interacted with a system using just jargon expressions and six with a system using descriptive expressions. For more discussions on our user simulation models and the corpus, please refer to [Janarthanam and Lemon, 2009b, Janarthanam and Lemon, 2009a, Janarthanam and Lemon, 2010b].

1.4. 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. 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.

1.4.1. Reward function

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 appropriate expressions for each referent $r$, with respect to the user’s knowledge. 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, a 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 $RA_r$, and then calculate the overall mean adaptation accuracy (AA) over all referents as shown below.

$$RA_r = \frac{\#(\text{appropriate\_expressions}(r))}{\#(\text{instances}(r))}$$
AdaptationAccuracy\( AA = \frac{1}{\#(r)} \sum_r RA_r \)

1.4.2. 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. The user simulation was calibrated to produce three types of users: Novice, Intermediate and Expert, randomly but with equal probability.

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. For instance, choosing a jargon expression could evoke a clarification request from the user, based on which, the dialogue manager updates the internal user model \( (UM,s) \) 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) during training. It can be seen in Figure 2 that towards the end the curve plateaus, signifying that learning has converged.

1.5. Evaluation

In this section, we present the details of the evaluation process, the baseline policy, the metrics used, and the results. In a recent study, 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 [Janarthanam and Lemon, 2010b]. An interesting issue for research in this area is to what extent evaluation results obtained in simulated environments transfer to evaluations with real users [Lemon et al., 2006].

1.5.1. Baseline system

In order to compare the performance of the learned policy with a baseline, a simple rule-based policy was built. This baseline was chosen because it performed better in simulation, compared to a variety of other baselines [Janarthanam and Lemon, 2010b]. It uses jargon for all referents by default and provides clarifications when requested. It exploits the user model in subsequent references after the user’s knowledge of the expression has been set to either yes or no. Therefore, although it is a simple policy, it adapts to a
certain extent (‘locally’). We refer to this policy as the ‘Jargon-adapt’ policy. It should be noted that this policy was built in the absence of expert domain knowledge and/or an expert-layperson corpus.

1.5.2. Experimental setup

We evaluated the two policies with real users. 36 university students from different backgrounds (e.g. Arts, Humanities, Medicine and Engineering) participated in the evaluation. 17 users were given a system with Jargon-adapt policy and 19 users interacted with a system with Learned-DS policy. Each user was given a pre-task recognition test to record his/her initial domain knowledge. The experimenter read out a list of technical terms and the user was asked to point out to 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 setup 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.

All users interacted with a wizarded system employing one of the two REG policies (see Figure 3). The user’s responses were intercepted by a human interpreter (or “wizard”) and were 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 policy used by the system. The respective policies chose only the referring expressions to generate the system utterance for the given dialogue action. The system utterances were converted to speech by a speech synthesizer (Cereproc) and were played to the user.
1.5.3. Metrics

In addition to the adaptation accuracy mentioned in Section 1.4.1, we also measure other parameters from the conversation in order to show how learned adaptive policies compare with other policies on other dimensions. We also measure the learning effect on the users as (normalised) learning gain ($LG$) produced by using unknown jargon expressions. This is calculated using the pre- and post-test scores for the user domain knowledge ($DK_u$) as follows.

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

Dialogue time (DT) is the actual time taken for the user to complete the task. We measured task completion (TC) by examining the user’s broadband setup after the task was completed (i.e. the percentage of correct connections that they had made in their final set-up).

1.5.4. Results

We compare the performance of the two strategies on real users using objective parameters and subjective feedback scores. Tests for statistical significance were performed using Mann-Whitney tests for 2 independent samples (due to non-parametric nature of the data).

Table 2 presents the mean accuracy of adaptation (AA), learning gain (LG), dialogue time (DT), and task completion (TC), produced by the two strategies. The Learned-DS strategy produced more accurate adaptation than the Jargon-adapt strategy ($p<0.0001$). Higher accuracy of adaptation (AA) of the Learned-DS strategy translates to less dialogue time ($p<0.001$) and higher task completion ($p<0.0001$) than the Jargon-adapt policy. However, there was no significant difference in learning gain (LG).

Table 3 presents how the users subjectively scored on a agreement scale of 1 to 4 (with 1 meaning ‘strongly disagree’), different features of the system based on their conversations with the two different strategies.
<table>
<thead>
<tr>
<th>Metrics</th>
<th>Jargon-adapt</th>
<th>Learned-DS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptation Accuracy (AA)</td>
<td>63.91</td>
<td>84.72 **</td>
</tr>
<tr>
<td>Learning Gain (LG)</td>
<td>0.71</td>
<td>0.74</td>
</tr>
<tr>
<td>Dialogue Duration (DT)</td>
<td>7.86</td>
<td>6.98 *</td>
</tr>
<tr>
<td>Task Completion (TC)</td>
<td>84.7</td>
<td>99.47 **</td>
</tr>
</tbody>
</table>

* Statistical significance (*p* < 0.001).
** Statistical significance (*p* < 0.0001).

Table 2: Evaluation with real users

<table>
<thead>
<tr>
<th>Feedback questions</th>
<th>Jargon-adapt</th>
<th>Learned-DS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1. Quality of voice</td>
<td>3.11</td>
<td>3.36</td>
</tr>
<tr>
<td>Q2. Had to ask too many questions</td>
<td>2.23</td>
<td>1.89</td>
</tr>
<tr>
<td>Q3. System adapted very well</td>
<td>3.41</td>
<td>3.58</td>
</tr>
<tr>
<td>Q4. Easy to identify objects</td>
<td>2.94</td>
<td>3.37 *</td>
</tr>
<tr>
<td>Q5. Right amount of dialogue time</td>
<td>3.23</td>
<td>3.26</td>
</tr>
<tr>
<td>Q6. Learned useful terms</td>
<td>2.94</td>
<td>3.05</td>
</tr>
<tr>
<td>Q7. Conversation was easy</td>
<td>3.17</td>
<td>3.42</td>
</tr>
<tr>
<td>Q8. Future use</td>
<td>3.22</td>
<td>3.47</td>
</tr>
</tbody>
</table>

* Statistical significance (*p* < 0.05).

Table 3: Real user feedback

Users’ feedback on different features of the systems were not very different from each other. However, users did feel that it was easier to identify domain objects with the Learned-DS strategy than the Jargon-adapt strategy (*p*<0.05). To our knowledge, this is the first study to show a significant improvement in real user ratings for a learned policy in spoken dialogue systems (normally, objective metrics show an improvement, but not subjective scores [Lemon et al., 2006]).

1.5.5. Analysis

The results show that the Learned-DS strategy is significantly better than the hand-coded Jargon-Adapt policy in terms of adaptation accuracy, dialogue time, and task completion rate. The initial knowledge of the users (mean pre-task recognition score) of the two groups were not significantly different from each other (Jargon-adapt = 7.35, Learned-DS = 6.57). Hence there is no bias on the user’s pre-task score towards any strategy. While the Learned-DS system adapts well to its users globally, the Jargon-adapt system adapted only locally. This led to higher task completion rate and lower dialogue time.

The Learned-DS strategy enabled the system to adapt using the dependencies that it learned during the training phase. For instance, when the user asked for clarification on some referring expressions (e.g. “ethernet cable”), it used descriptive expressions for domain objects like ethernet light and ethernet socket. Such adaptation across referents enabled the Learned-DS strategy to score better than the Jargon-adapt strategy.
strategy. Since the agent starts the conversation with no knowledge about the user, it learned to use information seeking moves (use jargon) at appropriate moments, although they may be inappropriate. But since it was trained to maximize the adaptation accuracy, the agent also learned to restrict such moves and start predicting the user’s domain knowledge as soon as possible. By learning to trade-off between information-seeking and adaptation, the Learned-DS policy produced a higher adaptation with real users with different domain knowledge levels.

The users however did not generally rate the two policies differently. However, they did rate it (significantly) easier to identify objects when using the learned policy. For the other ratings, users seemed to be not able to recognize the nuances in the way the system adapted to them. They could have been satisfied with the fact that the system adapted better (Q3). 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 et al., 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 is better. Results in table 3 lead us to conclude that perhaps users need to compare two or more strategies in order to judge the strategies better.

1.6. Conclusion

We presented new results from an evaluation with real users. In this study, we have shown that user-adaptive REG policies can be learned using an RL framework and data-driven user simulations. It learned to trade off between adaptive moves and information seeking moves automatically to maximize the overall adaptation accuracy. The learned policy started the conversation with information seeking moves, learned a little about the user, and started adapting dynamically as the conversation progressed. We also showed that the learned policy performs better than a reasonable hand-coded policy with real users in terms of accuracy of adaptation, dialogue time, task completion, and a subjective evaluation. Finally, this work provides further evidence that evaluation results obtained in simulated environments can transfer reliably to evaluations with real users [Lemon et al., 2006].

Whether the learned policy would perform better than a hand-coded policy which 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 behaviour and adapt accordingly. We also believe that this framework can be extended to include other decisions in NLG besides REG [Dethlefs and Cuayahuitl, 2010].
2. Testing TTS policies with real users

2.1. Introduction

The final CLASSiC System 3 was designed to test TTS speaking style variants. Each system prompt is available in 3 variants: “neutral”, “calm”, and “pushy”. CLASSiC System 3 includes the online Learning Manager module [Laroche et al., 2011], which has automatically adapted its policy during the evaluation campaign. Contrary to the case of the commercially deployed system which contained prompt wording variants, we were not able to observe a significant improvement of the system task completion rate in this experiment.

2.2. Raw performances

Figure 4 presents the evolution of the system task completion rate across the evaluation campaign according to both the system perception and the instructions given to the users. The horizontal axis gives the days where dialogues were collected. The vertical axis gives the percentage of task completion. The middle curve is a 3-day ($d-1,d,d+1$) average task completion rate. The upper and lower curves represent the 95% confidence interval.

The curve is very representative of the changes in the experimental conditions: during the experimentation the task completion rate was slightly rising. At the end of November 2010, the learned strategy was reset, to try and confirm the positive effect of selecting a TTS style.

Figure 4: Evolution of the raw system performance for System 3
2.3. Task completion rate on nominal conditions

As one can see in Figure 4, even on a 3-day average, the evolution of the task completion rate remains inside the uncertainty margin, and does not show a significant increase.

2.4. Internal and effective task completion rates

The online learning in System 3 is fed back with the system internal completion indicator. This indicator only reveals that the system has succeeded in taking an appointment, even if the booked slot is not available in the user agenda. Figure 5 shows how System 3 “thinks” it has performed. Note that the internal task completion rate is roughly 10% higher than the “real” system task completion rate. This shows that the effects of external influences are really important compared to the phenomenon we were trying to characterise.

![Graph showing internal system performance for System 3](image)

Figure 5: Evolution of the internal system performance for System 3

2.5. Perceived task completion rate

We have first observed a gap between “objective” task completion rate as can be computed by the system, and the “perceived” task completion rate as reported by the tester through a questionnaire. This gap raises two complementary issues and we had to consider whether it was really significant. If the differences were significant, we would have to question the use of learning algorithms based on “objective” indicators, which may be trying to optimise on the wrong criterion.

Figure 6 shows the gap, and its associated confidence intervals. The red bars represent the system perceived task completion rate with its upper and lower error margins. The blue bars represent the user
perceived task completion rate with its upper and lower error margins. One can clearly see that the differences are not significant. This means that the volume of data collected in this trial was insufficient to pursue further investigation of the current learning algorithm. To observe a statistically significant difference of 2% on the current level of performance (around 80%), one would need around 1600 samples in each class to be compared.

We would therefore like to raise the issue that analysis of such a fine-grained phenomenon on common real-world dialogue systems would require a much larger data collection, perhaps using systems on a national or international scale.

![Comparison between system and user perceived task completion rate](image)

**Figure 6: Comparison between system and user perceived task completion rate**

### 2.6. Conclusion for TTS in CLASSiC System 3

However cautious we were when trying to show the benefits of learning the best voice alternatives between carefully crafted TTS variants, we were not able to exhibit improvements as important as with the commercially deployed system, where the variants focussed instead on the wording (up to 10% improvement [Jurcicek et al., 2011, Putois et al., 2010]). The task completion rate of CLASSiC System 3 was already comparable with the task completion rate observed on the real 1013+ service after optimisation, and our expected progress margin was therefore already small.

Reinforcement learning methods have proven to be useful in some of the experiments with the commercial systems (e.g. when we use it in the commercial 1013+ system [Bretier et al., 2009, Laroche et al., 2010, Jurcicek et al., 2011, Putois et al., 2010]). However, these experiments were either theoretical, or on a very-large scale system receiving millions of calls, and we were not able to reproduce such benefits during the experimentation and evaluation of the TTS variants in CLASSiC System 3.
In summary, we propose three use cases. The first use case is using simulations to validate the correctness of the learning algorithms, as shown in previous work. The second use case is making evaluations with real users, which is prone to failure because of a lack of data. The last use case is making direct evaluations and monitoring on very large scale production systems, which has proved a success for the 1013+ service.
3. Summary and Conclusion

We have presented results from two different real-user evaluations of data-driven approaches to Natural Language Generation and Text-To-Speech. One evaluated user-adaptive referring expression generation (REG) in a Self-Help system, and the second concerned TTS voice variants for an Appointment Scheduling (AS) Spoken Dialogue System.

In the first case, we previously developed a reinforcement learning (RL) framework in which a spoken dialogue system learns REG policies which can adapt to unknown users online (published at ACL 2010: [Janarthanam and Lemon, 2010b]). For real users of such a system, we have shown here that (in comparison to an adaptive hand-coded baseline policy) the learned policy performs significantly better, with a 20.8% increase in adaptation accuracy, 11% decrease in time taken, and a 14.7% increase in task completion rate. An initial version of his work was published at SIGDIAL 2010 [Janarthanam and Lemon, 2010a]. The learned policy also has a significantly better subjective rating from users, in terms of the ease of identifying objects in the dialogue task. This is because the learned policies adapt online to changing evidence about the user’s domain expertise. We also discussed the issue of evaluation in simulation versus evaluation with real users.

The second evaluation tested an adaptive policy for selecting amongst carefully handcrafted TTS prompts. The evaluation was performed on CLASSiC System 3, which aimed at improving an Appointment Scheduling task. No significant improvement was observed in this task, and we discussed the possible causes of such a result.

(Note that a reinforcement learning approach to adaptive Temporal Expression Generation was also evaluated in the context of the overall evaluation of the CLASSiC “System 2” Appointment Scheduling system, see [Janarthanam et al., 2011]. This evaluation is therefore reported in deliverable 6.4. In addition, we also evaluated a trained NLG component for Information Presentation (recommendation) in the TownInfo system (“System 1”). This evaluation is also reported in D6.4, in the context of full system evaluations, and a publication is in preparation. See the Appendices for brief summaries of these evaluations.)
References


dm component, final prototype. Prototype D1.1.3, CLASSIC Project.


A. Appendix A: Evaluation of trained NLG in TownInfo (System 1)

CLASSiC System 1 was a VoIP system for searching real venues in Cambridge. One variant of the system (TownInfo+NLG) had a trained NLG component for recommendation [Rieser and Lemon, 2010], which was activated only when the system presented items to the user. Table 4 shows the effects of this component.

<table>
<thead>
<tr>
<th>Trial</th>
<th>System</th>
<th>NumDials</th>
<th>ObjSucc -Partial Comp (ci)</th>
<th>ObjSucc -Full Comp (ci)</th>
<th>ObjSucc -Goal Inferred (ci)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nov ’10</td>
<td>TownInfo</td>
<td>238</td>
<td>78.57 (5.56)</td>
<td>60.50 (6.28)</td>
<td>65.60 (5.89)</td>
</tr>
<tr>
<td></td>
<td>TownInfo+NLG</td>
<td>259</td>
<td>86.10 (5.03)*</td>
<td>62.16 (6.04)</td>
<td>64.03 (5.63)</td>
</tr>
<tr>
<td>Feb ’11</td>
<td>TownInfo</td>
<td>199</td>
<td>73.37 (6.14)</td>
<td>46.73 (6.93)</td>
<td>43.93 (6.54)</td>
</tr>
<tr>
<td></td>
<td>TownInfo+NLG</td>
<td>105</td>
<td>77.23 (8.02)</td>
<td>49.50 (9.56)</td>
<td>50.85 (8.81)</td>
</tr>
<tr>
<td>Feb ’11 MTurk</td>
<td>TownInfo</td>
<td>402</td>
<td>51.00 (4.39)</td>
<td>28.86 (4.43)</td>
<td>44.63 (4.64)</td>
</tr>
<tr>
<td></td>
<td>TownInfo+NLG</td>
<td>130</td>
<td>60.77 (8.39)*</td>
<td>37.69 (8.33)</td>
<td>47.68 (7.67)</td>
</tr>
<tr>
<td>Averages</td>
<td>TownInfo</td>
<td>839</td>
<td>67.65</td>
<td>51.39</td>
<td>45.36</td>
</tr>
<tr>
<td></td>
<td>TownInfo+NLG</td>
<td>494</td>
<td>74.70</td>
<td>54.19</td>
<td>49.78</td>
</tr>
</tbody>
</table>

Table 4: NLG evaluation for TownInfo (System 1): objective success rates. (* = p < 0.05)

In the Nov ’10 and Feb’11 MTurk evaluations, the partial completion score for the TownInfo +NLG system (i.e. including the trained NLG component) was significantly higher than the TownInfo system (p=0.02, z-test), suggesting that the more elaborate venue offers from the trained NLG component helped the user find the venue they were looking for more easily. Publication of these results is planned.

(Please see deliverable D6.4 for full presentation and discussion.)

B. Appendix B: Evaluation of trained NLG in Appointment Scheduling System (System 2)

During this evaluation, we compared two types of NLG methods for generating Temporal Referring Expressions (TRE) for appointment dates. See [Janarthanam et al., 2011] for details. A data-driven policy developed at Heriot-Watt and Edinburgh (see also Deliverable D5.4) was activated when the system informs the user of an available time slot. This system was compared to the exact same system but with a rule-based adaptive baseline system (developed at the University of Cambridge). In this rule-based policy, MONTH, DATE, and TIME were always absolute, DAY was relative if the target date was less than three days away (i.e. “today, tomorrow, day after tomorrow”), and WEEK was always relative (i.e. “this week, next week”). All 5 information units were included in the realisation (e.g. “Thursday the 15th July in the afternoon, next week”) although the order was slightly different (DAY-DATE-MONTH-TIME-WEEK).

Results from the System 2 NLG study are summarised in Table 5. The data-driven NLG policy (‘Learned TRE’) showed significant improvement in Perceived Task Success (+23.7%) although no significant difference was observed between the two systems in terms of Actual Task Success (Chi-square test, df=1). Overall user satisfaction (the average score of all the questions) was also significantly higher (+5%)². Dialogues with the learned policy were significantly shorter with lower Call Duration in terms of time (-15.7%)² and average words per system turn (-23.93%)².

Version: 1.0 (final) Distribution: Public
<table>
<thead>
<tr>
<th></th>
<th>Learned TRE</th>
<th>Baseline TRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual Task Success</td>
<td>80.05%</td>
<td>78.57%</td>
</tr>
<tr>
<td>Perceived Task Success</td>
<td>74.86%*</td>
<td>60.50%</td>
</tr>
<tr>
<td>User satisfaction</td>
<td>4.51*</td>
<td>4.30</td>
</tr>
<tr>
<td>No. system turns</td>
<td>22.8</td>
<td>23.2</td>
</tr>
<tr>
<td>Words per system turn</td>
<td>13.16*</td>
<td>17.3</td>
</tr>
<tr>
<td>Call duration (seconds)</td>
<td>88.60 *</td>
<td>105.11</td>
</tr>
</tbody>
</table>

Table 5: System 2 Appointment Scheduling NLG evaluation: Results with real users (* = statistically significant differences at p<0.05)

In summary, these results show that using an NLG policy trained on data results in shorter dialogues and greater confidence in the user that they have had a successful dialogue.

(Please see deliverable D6.4 and [Janarthanam et al., 2011] for full presentation and discussion.)

\(^2\)independent two-tailed t-test p < 0.05