Reinforcement learning for parameter estimation in statistical spoken dialogue systems

Filip Jurčíček, Blaise Thomson, Steve Young

Cambridge University, Engineering Department, Trumpington Street, Cambridge, CB2 1PZ, UK

Abstract

Reinforcement techniques have been successfully used to maximise the expected cumulative reward of statistical dialogue systems. Typically, reinforcement learning is used to estimate the parameters of a dialogue policy which selects the system's responses based on the inferred dialogue state. However, the inference of the dialogue state itself depends on a dialogue model which describes the expected behaviour of a user when interacting with the system. Ideally the parameters of this dialogue model should be also optimised to maximise the expected cumulative reward.

This article presents two novel reinforcement algorithms for learning the parameters of a dialogue model. First, the Natural Belief Critic algorithm is designed to optimise the model parameters whilst the policy is kept fixed. This algorithm is suitable, for example, in systems using a handcrafted policy, perhaps prescribed by other design considerations. Second, the Natural Actor and Belief Critic algorithm jointly optimises both the model and the policy parameters. The algorithms are evaluated on a statistical dialogue system modelled as a Partially Observable Markov Decision Process in a tourist information domain. The evaluation is performed with a user simulator and with real users. The experiments indicate that model parameters estimated to maximise the expected reward function provide improved performance compared to the baseline

Email addresses: fj228@eng.cam.ac.uk (Filip Jurčíček), brmt@eng.cam.ac.uk (Blaise Thomson), sjy@eng.cam.ac.uk (Steve Young)

handcrafted parameters.

Keywords: spoken dialogue systems, reinforcement learning, POMDP, dialogue management

---

1. Introduction

Spoken dialogue systems enable human users to use their voice as a communication medium when interacting with machines. In recent years, it has been argued that Partially Observable Markov Decision Processes (POMDPs) provide a principled way to model uncertainty in statistical spoken dialogue systems (Roy et al., 2000; Young, 2002; Young et al., 2010). A POMDP dialogue manager includes three main parts: a dialogue model representing state information such as the user's goal, the user's last dialogue act and the dialogue history; a policy which selects the system's responses based on the inferred dialogue state; and a cumulative reward function which specifies the desired behaviour of the system. In a POMDP system, the dialogue model provides a compact representation for the distribution of the unobserved dialogue state called the *belief state* and it is updated every turn based on the observed user inputs in a process called *belief monitoring*. Exact belief monitoring of the full dialogue state is intractable for all but the simplest systems. One way to address this issue is to represent the state in the compact and approximate form of a dynamic Bayesian Network (BN) (Thomson and Young, 2010). Another way is to maintain probabilities only for the most likely dialogue states. To achieve efficient belief monitoring, indistinguishable states can be grouped into partitions and consequently the belief monitoring is performed on partitions instead of the individual states (Young et al., 2010). In in both cases, the dialogue model is parameterised and a substantial improvement in performance can be achieved by optimising these parameters.

The policy selects the dialogue system's responses (actions) based on the belief state at each turn, and it is typically trained using reinforcement learning with the objective of maximising the cumulative reward function. The choice

of cumulative reward function is a dialogue design issue, but it will typically provide positive rewards for satisfying the user's goal, and negative rewards for failure and wasting time. Ideally both the dialogue model and the policy should be designed to maximise the cumulative reward function.

This article presents two novel reinforcement algorithms for learning the parameters of a dialogue model. The first algorithm called Natural Belief Critic (NBC) aims to learn the model parameters while the dialogue policy remains fixed. This algorithm is mainly suitable for the optimisation of dialogue systems which depend on handcrafted policies, perhaps constrained by other design considerations such as costumers requirements. The NBC algorithms is based on policy gradient methods (Peters et al., 2005). However, generalisation of these methods was necessary since when using policy gradient methods, derivatives of the expected reward function with respect to all parameters of the dialogue system are required. Since scalable dialogue systems usually depend on some form of summary space compression (Williams and Young, 2005), this is rarely possible. The difficulty with using a summary space is that the mapping from the original space to the summary space depends on the dialogue model parameters through a handcrafted function extracting non-continuous features from the belief state. Consequently, closed form derivatives of the dialogue model parameters are not available. However, this problem can be alleviated by assuming that the dialogue model parameters come from a prior distribution that is differentiable with respect to its parameters.

When joint optimisation of both the dialogue model parameters and the policy parameters is required, previous work (Jurčíček et al., 2010) suggested using interleaved training using the NBC algorithm for the dialogue model and then the Natural Actor Critic (NAC) algorithm (Peters et al., 2005) for policy training. However, this interleaved training procedure is applicable only when a reasonable initial set of handcrafted model parameters is available. More recent work (Jurčíček et al., 2011b) proposed the Natural Actor and Belief Critic (NABC) algorithm which is a more general and mathematically founded approach to the joint optimisation of the model and the policy parameters. The

main idea is that the policy does not have to be fixed and that the NBC and NAC algorithms can be naturally combined into one joint optimisation method. As a result, this article will describe and evaluate later approach.

The main contribution of this article is to present an extended and unified description of the NBC and NABC algorithms and to provide a full evaluation with real users. In the previous works (Jurčíček et al., 2010, 2011b), the NBC and NABC algorithms were evaluated with a simulated user. Although it is an efficient way test the performance of spoken dialogue systems, it is not always a good predictor of performance with real human users. Possible problems connected with evaluating dialogue systems on a user simulator instead of human users include a potential discrepancy in user cooperativeness, expertise and in handling dialogue misunderstandings (Schatzmann, 2008). Also the quality of simulation of speech recognition and spoken language understanding errors can greatly affect the accuracy of the simulations. Therefore evaluation with real users is important to validate the results obtained with a user simulator. The methods are presented and evaluated in the context of the BUDS POMDP dialogue manager (Thomson and Young, 2010) which uses a dynamic Bayesian Network to represent the dialogue state. However, the methods are sufficiently general that they could be used to optimise virtually any parameterised dialogue model. The article is structured as follows. Section 2 briefly describes the BUDS dialogue manager and the method it uses for policy representation (Thomson and Young, 2010). Section 3 then describes policy gradient methods and a specific form called the Natural Actor Critic (NAC) algorithm which is used to optimise the BUDS policy. In Section 4, the proposed Natural Belief Critic algorithm is presented as a generalisation of the NAC algorithm. Section 5 discusses the original interleaved training procedure for joint optimisation of the model and the policy parameters which is then followed by a description of the Natural Actor and Belief Critic algorithm. The algorithms are evaluated on a system designed for the tourist information domain in Section 6. Finally, Section 7 discusses alternative methods and Section 8 concludes the article.

2. BUDS dialogue manager

In a POMDP dialogue system, the true dialogue state s_t is unknown. Therefore, the dialogue policy selects an action a_t at time t based on the distribution over all states called the belief state, b_t. The estimate of the belief state depends on the past and current observations, $o_1, \ldots, o_t$, and actions, $a_0, \ldots, a_{t-1}$, which are referred to as the observed history:

$$h_t = \{a_0, o_1, \ldots, a_{t-1}, o_t\}\,. \tag{1}$$

If the system is Markovian then the belief state b_t depends only on the previous belief state b_{t-1}, the current observation o_t and the last system action a_{t-1}. Using Bayes theorem, the belief state b_t can be computed as follows:

$$\begin{aligned} b_t &= b(s_t|h_t;\tau) \\ &= k \cdot p(o_t|s_t;\tau) \sum_{s_{t-1}} p(s_t|a_{t-1}, s_{t-1};\tau) b(s_{t-1}|h_{t-1};\tau) \end{aligned} \tag{2}$$

where the transition probability $p(s_t|a_{t-1}, s_{t-1};\tau)$ and the observation probability $p(o_t|s_t;\tau)$ represent the dialogue model which is parameterised by τ and k is a normalisation constant. The dialogue model parameters τ are described in more detail for a particular instance of a dialogue model in Section 2.2.1.

2.1. The observations and system actions

In a spoken dialogue system, the observations and the system actions are represented by dialogue acts. In BUDS, a dialogue act conveys the user or system intention (such as inform, request, etc) and a list of slot-value pairs (e.g. type=hotel, area=east). In some cases, the value can be omitted, for example, where the intention is to query the value of a slot e.g. "request(food)". When there is uncertainty in the speech understanding process, the input to the dialogue manager is a list of alternative dialogue acts. For example, the utterance "I want an Indian restaurant in the cheap price range" spoken in a noisy background might yield

`inform(food=Indian, type=restaurant, pricerange=cheap)` {0.6}
`inform(food=Italian, type=restaurant)` {0.4}

where the numbers in the brackets represent the probabilities of each dialogue act. A full description of the dialogue act set used by the BUDS system is given in (Young, 2007). An example of a typical dialogue in the Cambridge Restaurant Information domain is shown in Table 1.

System	Thank you for calling the Cambridge Information system. `hello()`
User	Hi, I'm looking for an expensive Chinese restaurant. `inform(type=restaurant,food=Chinese,pricerange=expensive)`
System	There is no place to eat serving Chinese food in the expensive price range. `inform(name=none,pricerange=expensive)`
User	I want something cheap then. `inform(pricerange=cheap)`
System	Yu Garden serves Chinese food. It is in the cheap price range. `inform(name='Yu Garden',pricerange=cheap)`
User	Ok, give me the address. `request(addr)`
System	Yu Garden is located at 529 Newmarket Road. `inform(name='Yu Garden',addr='529 Newmarket Road')`
User	Thank you. Goodbye. `bye()`

Table 1: An example of a dialogue from the Cambridge Restaurant Information domain.

2.2. The dialogue model

A naive implementation of the belief update equation (2) is not tractable since there are billions of states in a real-world spoken dialogue system. Thus,

the BUDS dialogue manager uses a Bayesian Network (BN) to represent the state of the POMDP dialogue system, where the network is factored according to the slots in the system (Thomson and Young, 2010). This factoring is straightforward for so-called *slot filling* applications where the complete dialogue state can be reduced to the state of a small number of slots that require to be filled (Williams and Young, 2007a,b). For more complex applications, the assumption of independence between slots can be relaxed somewhat by using dynamic Bayesian Networks (Thomson et al., 2008a,b). Provided that each slot or network node has only a few dependencies, tractable systems can be built and belief estimates maintained with acceptable accuracy using approximate inference (Bishop, 2006).

The BUDS system's inference algorithm uses grouped loopy-belief propagation (Thomson and Young, 2010). It maintains marginal belief estimates only for slot values which were mentioned by the user while those values which were not mentioned are grouped assuming that they have the same probability within the group. Grouped loopy-belief propagation reduces computation time significantly in situations where the number of potential slot values is in the hundreds but the user mentions only a few of them.

2.2.1. Example: The Cambridge Restaurant Information system

The task of the Cambridge Restaurant Information system (CamInfo) is to aid tourists in finding restaurants and to give information about the venue. In CamInfo, the dialogue state is factored into three components: the user goal g, the user action u and the dialogue history d. In addition, the goal and the history are further factored into subgoals and subhistories according to a set of slots in the system.

The BN contains nine subgoals: name of the venue, type of venue, area, price range, nearness to a particular location, type of drinks, food type, number of stars and type of music. Every subgoal has a corresponding subhistory node. The subhistory nodes allow the system designer to store information about whether a user requested information or the system informed the user

about some slot. The applicable values for a subhistory node are "nothing-said", "user-requested", and "system-informed." The network also has nodes to represent address, telephone number, a comment on the venue and the price. However, for these only their subhistory nodes are used since a user can only ask for values of these slots and cannot specify them as query constraints. Finally, the network has two special nodes. The "method" node stores the probability that the user is searching for a venue by constraints (represented by the value "byconstraint") or by name ("byname"), or the user is searching for an alternative venue ("alternative"), or the user wants to restart ("restart") or end the dialogue ("bye"). The "discourse" node infers whether the user wants the system to repeat the last system action ("repeat"), provide more information about the last offered venue ("more"), acknowledge the last system dialogue act (ack), or greet the system (hello). Although the dialogue manager does not ask about these nodes explicitly, the probabilities of their values are inferred just like in any other node.

The user action u is the estimate of the true dialogue act given the observation o. To compute this, some conditional independence assumptions are made: the user act depends on all subgoals and the last system action; the subgoals depend on their previous value, the last system action and may depend on any combination of other current subgoals; the subhistory depends on the previous subhistory and the current user act. Figure 1 shows a part of the resulting Bayesian network for two time-slices of the system. Note that the user subgoals and the user act are assumed to be conditionally independent of the subhistories. The network also includes observed system actions a and user dialogue acts, o.

The BN model parameters τ comprise the set of conditional probabilities of the node values. For example, the "food" subgoal values are described by the probability $p(g''_{food}|g'_{food}, g''_{type}, a'; \tau_{food})$ parameterised by τ_{food}. To reduce the number of parameters specifying the distributions in the subgoals, some parameters are tied together on the assumption that the probability of change in the subgoals is constant given the last system action and the parent subgoal. For

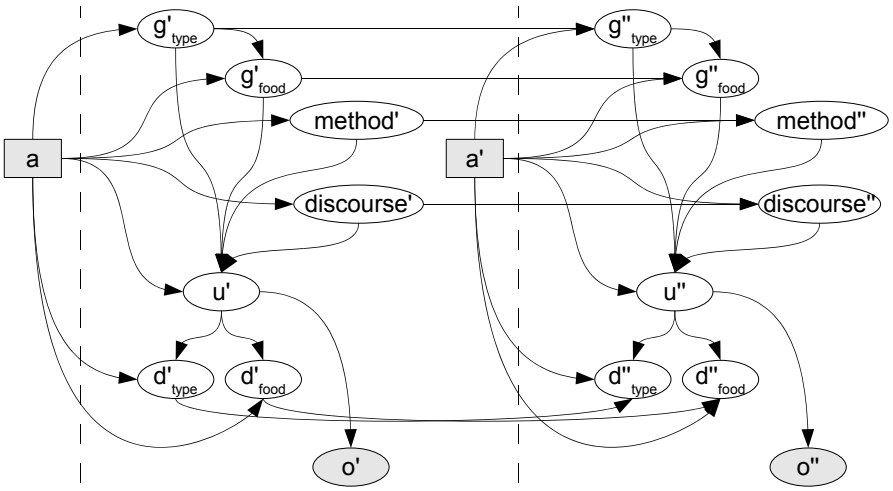

Figure 1: An example factorisation for the Bayesian network representing part of the CamInfo tourist information system. The white nodes are unobserved variables and the grey nodes are observed variables. The squares denote actions decided by a policy.

example, the probability of change from "Chinese" to "Indian" in the subgoal "food" is equal to the probability of change from "Chinese" to "Italian". As a result, instead of having N^2 parameters, where N is the number of values in the subgoal, only a single probability of change $\tau_{food,change}$ is needed (the probability of the subgoal staying the same is equal to $1 - \tau_{food,change}$). The subhistory nodes do not use parameter tying as they have only three values. The history, "method", and "discourse" nodes use fully parameterised conditional probabilities in order to capture the detailed characteristics of dialogue flow.

Overall this results in a dialogue model with 565 parameters.

2.3. The policy

The BUDS dialogue manager uses a stochastic policy $\pi(a|b;\theta)$ which gives the probability of the system taking action a given belief state b and policy parameters θ. When used in the dialogue manager, the policy distribution is sampled to yield the required action at each turn. To reduce complexity, for every action a, the belief state is mapped into a vector of features, $\Phi_a(b)$ and

the policy is then approximated by a softmax function:

$$\pi(a_t|b(\cdot|h_t;\tau);\theta) \approx \frac{e^{\theta^T \cdot \Phi_{a_t}(b(\cdot|h_t;\tau))}}{\sum_{\tilde{a}} e^{\theta^T \cdot \Phi_{\tilde{a}}(b(\cdot|h_t;\tau))}}. \quad (3)$$

To estimate the policy parameters, BUDS uses the episodic Natural Actor Critic (NAC) algorithm (Peters et al., 2005). BUDS also supports handcrafted policies (for more details see Section 2.3.3).

2.3.1. Summary actions and mapping summary actions back to dialogue acts

A further reduction in complexity can be achieved by utilising summary actions (Thomson and Young, 2010). For example, if the dialogue manager confirms the value of some subgoal then it should always confirm the most likely value. As a result, the full set of actions is not needed. Once a summary action is proposed by the policy then it is mapped into a dialogue act by a heuristic algorithm based on the information in the belief state and the database.

The BUDS CamInfo dialogue manager, supports two types of summary actions: slot-level and global. The purpose of the slot-level summary actions is to prompt the user about a particular subgoal. They are defined for each subgoal in the domain except for the subgoal "name". The system is not allowed to ask a user for the name of a venue since the goal is to find a suitable venue and provide the name. There are three slot-level summary actions: request, confirm, and select. The following list details the slot-level summary actions and how they are transformed into full dialogue acts:

- request subgoal X - requests information about subgoal X from a user. It is transformed into the full dialogue act "request(X)". No information from the database or the belief state is necessary.

- confirm subgoal X - asks a user to confirm the most likely value in the subgoal X. The heuristics transform the action into the full dialogue act "request(X=Y)", where Y is the most likely value in the subgoal X as obtained from the belief state.

- select subgoal X - prompts a user to select between the two most likely values for the subgoal X. It is mapped to the full dialogue act "select(X=Y, X=Z)" where the most likely value Y and the second most likely value Z in the subgoal X are obtained from the belief state.

Given the 8 subgoals (excluding the "name" subgoal) in the domain, there are 24 slot-level summary actions in total.

The global summary actions implement other actions which cannot be associated only with one subgoal. In the CamInfo, there are 10 global summary actions:

- inform about a venue - provides the name and properties of the venue matching the user's constraints inferred during the course of the dialogue. The summary action is mapped into the full dialogue act "inform(name=X, S1=V1, S2=V2, ...)" where X is the name of the venue being offered, and Sx and Vx are the provided constraints. For example, if the user wants a cheap English restaurant then the mapping heuristics provide not only the name of a matching venue but also the values for the price, food type and venue type. Given the previous example, the generated dialogue act would be the form "inform(name="The Axe", price=cheap, food=English, type=restaurant)."

- inform more - provides additional information about the last offered venue. The additional information is comprised of values for subgoals that were not provided by the user as constraints. For example, if the user wants a cheap restaurant and requests more information then the heuristic mapping generate an inform act providing more details about food type, location, opening hours, etc.

- inform requested slots - informs about all slots that were requested by the user. The user can request information for one or more slots at the same time or in multiple turns. The information about whether the user requested a particular slot is retrieved from its subhistory node.

For example, if the user requests information for the area and the opening hours then the heuristics generates "inform(name=N, area=A, openinghours=O)" where N is the name of the last offered venue, and the area A and the opening hours O are retrieved from the database.

- inform confirmed slots - informs about all slots that are being confirmed by the user. For example, if the user wants to confirm that the offered restaurant is cheap, "confirm(pricerange=cheap)", then the summary action is mapped into the full dialogue act "inform(name=X, pricerange=cheap)" if the venue X is indeed in the cheap pricerange. If the venue X is not in the cheap price range then it is mapped into "inform(name=X, pricerange=Y)" where Y is the correct value for the pricerange of the venue X.

- inform about alternatives - informs about an alternative venue which matches as many of the user's constraints as possible. Once an alternative venue is found then it is mapped in the same way as the summary action "inform about a venue."

- inform by name - informs about a venue specified by name. This action is used when the user asks "request(phone, name=The Bakers)."[1] The mapping heuristics generate an inform act providing the requested slots.

- restart - maps into the full dialogue act "restart()". When processed by the dialogue manager, it changes the belief state into the initial state and all previous information provided by the user is forgotten.

- repeat - repeats the last system's dialogue act.

- request more - maps into the full dialogue act "reqmore()" which asks a user for additional information.

- bye - says bye to the user and terminates the dialogue.

[1] "Can you give me a phone number of The Bakers?"

Note that the mapping of the summary actions "restart", "repeat", "request more", and "bye" is straightforward as they do not use any information from the belief state or the database.

Sometimes some of the summary actions are not valid in the current context of a dialogue. For example, the system cannot provide additional information about the last offered venue (the summary action "inform more") until some venue has been offered. As a result, the policies are required to propose only valid actions. In the case of a handcrafted policy, it can be designed to produce only valid summary actions. For a stochastic policy, if an invalid summary actions is sampled then it is detected by a simple heuristic and a new action is resampled.

2.3.2. Policy features

Referring again to the policy function (3), there are a variety of possible forms for the feature function Φ (Sutton and Barto, 1998). However, it is usually beneficial if it performs some form of tiling. The BUDS dialogue manager uses a slot-level grid-based approximation. First, it is assumed that the slot-level summary actions depend mainly on the probability distribution for that particular slot. Then, for every slot-level action a set of binary features is generated based on the probabilities of two most likely values in the corresponding subgoal and a predefined set of grid points. In BUDS, the binary features indicate which grid point is closest to the tuple formed from probabilities of the two most likely values. For example, there are 7 grid points in BUDS: $(1.0, 0.0)$, $(0.8, 0.0)$, $(0.8, 0.2)$, $(0.6, 0.0)$, $(0.6, 0.2)$, $(0.6, 0.4)$, $(0.0, 0.0)$. If the tuple of the two probabilities is $(0.4, 0.4)$ then the corresponding binary feature set is $\{0, 0, 0, 0, 0, 0, 1, 0\}$, where 1 indicates the index of the closest grid point (Thomson, 2010). In addition, there are features for the global actions. For example, they include the number of grounded slots in the BN or the number of matching venues. Overall there are 893 mostly binary features in the CamInfo domain. Consequently, the same number of weight parameters has to be learned for the stochastic policy.

2.3.3. Handcrafted policy

BUDS also supports handcrafted policies which are designed by an expert. These policies deterministically choose which action to take given the belief state and they have the form of if/then statements written in the source code of the dialogue manager.

In the CamInfo domain, the if/then statements are organised into an ordered sequence where the first statement is verified first. If the condition of the statement is satisfied then the corresponding action is executed. If the statement is not applicable then the handcrafted policy continues with next statement, and so forth. The following list enumerates the handcrafted policy for the CamInfo domain.

1. request values for slots in which the most likely value has probability lower than 0.3,
2. confirm the most likely values in slots where the most likely value has probability between 0.3 and 0.9,
3. select between the two most likely values if the sum of the probabilities for these values is more than 0.8,
4. inform about a venue if there is only one venue matching the provided constraints and the probability of the "byconstraint" value in the "method" node is more than 0.5,
5. inform about a venue specified by the "name" subgoal if the probability of the "byname" value in the "method" node is higher than 0.5,
6. inform about alternatives if the probability of the "alternative" value in the "method" node is higher then 0.5,
7. restart the dialogue if the probability of the "restart" value in the "method" node is higher than 0.5,
8. end the dialogue if the probability of the "bye" value in the "method" node is more than 0.5,
9. inform about slots requested by a user if the probability of the "user-requested" value of the corresponding subhistory node is more than 0.5,

10. repeat the last system's dialogue act if the probability of the "repeat" value in the "discourse" node is more than 0.5,
11. provide more (the "inform more" summary action) information about the last offered venue if the probability of the "more" value in the "discourse" node is more than 0.5,
12. request more information from the user.

3. Policy gradients

The goal of reinforcement learning is to maximise the expected cumulative reward J:

$$J(\theta, \tau) = E\left[\sum_{t=0}^{T-1} r_t | \pi_{\theta,\tau}\right],$$

where θ and τ are the parameters of the policy and the dialogue model respectively, and r_t is the reward assigned to the dialogue system at time t. Let H_t be the trajectory[2] of the dialogue system which consists of the observed history (1) together with the unobserved dialogue states:

$$H_t = \{s_0, a_0, o_1, s_1, \ldots, a_{t-1}, o_t, s_t\}.$$

Let $R(H)$ be the expected reward accumulated along the trajectory H and $p(H; \theta, \tau)$ be the probability of the trajectory H given the policy and dialogue model parameters. Then, the objective function can be expressed as the expected reward over all trajectories:

$$J(\theta, \tau) = \int p(H; \theta, \tau) R(H) dH. \tag{4}$$

Learning θ and τ can be achieved by a gradient ascent which iteratively adds a multiple of the gradient to the parameters being estimated. Using "the log likelihood-ratio trick" (Williams, 1992) and Monte Carlo approximation, the gradient can be estimated as follows:

$$\nabla J(\theta, \tau) = \frac{1}{N} \sum_{n=1}^{N} \nabla \log p(H_n; \theta, \tau) R(H_n), \tag{5}$$

[2]Note that the trajectory is sometimes called the *path* or the *complete history*.

where the sampled dialogues are numbered $n = 1, \ldots, N$.

By observing that the probability $p(H; \theta, \tau)$ is the product the probabilities of all actions, observations and state transitions along the trajectory H, the probability of the trajectory can be defined as:

$$p(H; \theta, \tau) = p(s_0) \prod_{t=1}^{T} p(o_t|s_t) p(s_t|a_{t-1}, s_{t-1}) \pi(a_{t-1}|b(\cdot|h_{t-1}; \tau); \theta), \tag{6}$$

where $p(s_0)$ is the initial state distribution. Since the true state observation and transition probabilities do not depend on θ and τ, the log-gradient can be written in the form:

$$\nabla \log p(H; \theta; \tau) = \sum_{t=0}^{T-1} \nabla \log \pi(a_t|b(\cdot|h_t; \tau); \theta).$$

Substituting this into (5) gives:

$$\nabla J(\theta, \tau) = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=0}^{T_n - 1} \nabla \log \pi(a_t^n|b(\cdot|h_t^n; \tau); \theta) R(H_n). \tag{7}$$

Note that the gradient now depends only on observed variables.

To obtain a closed form solution of (7), the policy π must be differentiable with respect to θ and τ. Conveniently, the softmax function (3) is differentiable with respect to θ. However, the policy is not usually differentiable with respect to τ - the dialogue model parameters. The difficulty is that the function Φ, which extracts features from the belief state, is most often a handcrafted function of non-continuous features. This problem also arises when the policy is handcrafted as described in Section 2.3.3. A handcrafted policy effectively prevents computing any gradients of the parameters in the dialogue system.

Although (7) cannot be used to optimise the model parameters, it can still be used to estimate the policy parameters. However, it has been shown that the use of the "plain" gradient given by (7) yields rather poor convergence properties compared to the natural gradient $\widetilde{\nabla} J(\theta) = F^{-1}(\theta) \nabla J(\theta)$ which is more effective for optimisation of statistical models where $F(\theta)$ is the Fisher Information Matrix (Amari, 1998). Based on this idea, Peters et al. (2005) developed a family of Natural Actor Critic algorithms which estimates a *natural gradient*

of the expected reward function. Appealing feature of the these algorithms is that in practice the Fisher Information Matrix does not need to be explicitly computed. In this work, the episodic version the Natural Actor Critic algorithm (NAC) is used to train the BUDS's stochastic policy. To obtain the natural gradient, w, of $J(\theta)$, NAC uses a least squares method to solve the following set of equations:

$$r_n = \left[\sum_{t=1}^{T_n} \nabla \log \pi(a_t^n | b_t^n; \theta)^T \right] \cdot w + C \quad \forall n \in \{1, \ldots, N\},$$

where r_n is the reward observed at the end of each dialogue n. Once w has been found, the policy parameters can be iteratively improved by $\theta' \leftarrow \theta + \beta w$, where β is a step size. There are many ways how to determine the step size β. In this work, β is manually set to a fixed value based on some initial experiments. The actual value used for the step size depends on the definition of the reward function as the size of the gradient scales linearly with the reward function.

Of all the policy optimisation algorithms tested with BUDS, the NAC algorithm has proved to be the most robust suggesting that the use of the natural gradient is critical. The question therefore arises whether this type of policy gradient method can be generalised to optimise not just the policy but the parameters of the dialogue model as well.

4. Learning of the dialogue model parameters

4.1. Natural Belief Critic algorithm

The Natural Belief Critic (NBC) algorithm is an extension of policy gradient methods, specifically the NAC algorithm, aimed at learning the dialogue model parameters, τ. However, the non-differentiable parameters τ of the objective function (4) are not estimated directly. Instead, the NBC algorithm assumes that τ come from a prior distribution $p(\tau; \alpha)$ and the prior is designed in a such way that it is differentiable with respect to the parameters α. Consequently, the prior parameters α become differentiable parameters of the objective function and the previously outlined policy gradient methods can be used to optimise α.

In each iteration, the algorithm samples multiple model parameters from the prior, evaluates the sampled parameters, and uses the observed rewards to update the prior parameters. After several iterations, the expected values for τ given the distribution $p(\tau;\alpha)$ provide new estimates for τ. The goal of NBC is to learn the parameters α of the prior distribution of the model parameters which maximise the expected cumulative reward. The algorithm assumes that the policy is fixed and only improvements to the parameters α are considered. For that reason in the rest of this section, the objective function will be denoted as $J(\alpha)$ and the policy parameters θ will be dropped from the equations.

An example of a practical case in which the NBC algorithm is applicable is a dialogue system with a handcrafted policy using a parameterised dialogue model. This can be of particular interest in commercial applications where greater control over the dialogue system behaviour is required.

The derivation of the algorithm follows the approach to policy gradients described in Section 3. Let the dialogue model parameters τ be sampled at the beginning of each dialogue, then τ becomes an observed part of the trajectory: $\tilde{H}_t = \{\tau, a_0, s_0, o_1, \ldots, a_{t-1}, s_t, o_t\}$. The probability of a trajectory $\tilde{H}$ is then defined as follows:

$$p(\tilde{H};\alpha) = p(s_0)p(\tau;\alpha)\prod_{t=1}^{T} p(o_t|s_t)p(s_t|a_{t-1},s_{t-1})\pi(a_{t-1}|b(h_{t-1};\tau))$$

Note that the probability of a trajectory $\tilde{H}$ depends on α instead of τ.

Now, a derivative of $J(\alpha)$ with respect to α can be computed. Similar to the result in (7), the solution to $\nabla J(\alpha)$ takes advantage of the independence of most of the factors in $p(\tilde{H};\alpha)$ on α after taking the logarithm. As a result, the Monte Carlo estimate of the gradient is:

$$\nabla J(\alpha) = \frac{1}{N}\sum_{n=1}^{N} \nabla \log p(\tau_n;\alpha) R(\tilde{H}_n). \tag{8}$$

If the prior for τ is chosen conveniently, then a closed form solution can be found. For more details see Section 4.2.

To improve the estimation of the gradient, a constant baseline, B, can be introduced into the equation above. Williams (1992) showed that the baseline

does not introduce any bias into the gradient. Nevertheless, if it is chosen appropriately it lowers the variance of the gradient. As result, the gradient is defined as:

$$\nabla J(\alpha) = \frac{1}{N} \sum_{n=1}^{N} \nabla \log p(\tau_n; \alpha)(R(\tilde{H}_n) - B). \tag{9}$$

The gradient defined in (9) still cannot be used as it includes the expected reward $R(\tilde{H}_n)$ which is not usually available. Typically, $R(\tilde{H}_n)$ is approximated by a function which is compatible with the distribution $p(\tau; \alpha)$ (Konda and Tsitsiklis, 2000; Sutton et al., 2000). A compatible function approximation of $R(\tilde{H}_n)$ parameterised by a vector w is as follows:

$$R(\tilde{H}_n) \approx R(\tilde{H}_n; w) = \nabla \log p(\tau_n; \alpha)^T \cdot w + C \tag{10}$$

To compute the parameters w, a least squares method can be used. In this case, the expected reward $R(\tilde{H}_n)$ is replaced by the observed reward r_n and the following set of equations is solved while minimizing the sum of the squares of the errors:

$$r_n = \nabla \log p(\tau_n; \alpha)^T \cdot w + C \quad \forall n \in \{1, \ldots, N\} \tag{11}$$

Once the approximation of the expected reward is defined, it can be applied to (8) to obtain the gradient $\nabla J(\alpha)$:

$$\nabla J(\alpha) \approx \frac{1}{N} \sum_{n=1}^{N} \nabla \log p(\tau_n; \alpha) \nabla \log p(\tau_n; \alpha)^T \cdot w \tag{12}$$

Note that the constant C was cancelled by an optimal constant baseline B which lowers the variance of the estimate (Williams, 1992).

However, it can be shown that the vector w is already the natural gradient which can be be used to iteratively improve the dialogue model parameters: $\alpha' \leftarrow \alpha + \beta w$, where β is a step size. This follows since

$$\frac{1}{N} \sum_{n=1}^{N} \nabla \log p(\tau_n; \alpha) \nabla \log p(\tau_n; \alpha)^T$$

is in fact an estimate of the Fisher Information matrix. Thus, equation (12) can be written as $\nabla J(\alpha) \approx F(\alpha) w$. As a result, the natural gradient of the expected

cumulative reward is:

$$\widetilde{\nabla} J(\alpha) = F^{-1}(\alpha)\nabla J(\alpha) \approx F^{-1}(\alpha)F(\alpha)w = w \tag{13}$$

Similar to the approach in Section 3, the step size β is manually tuned. Note that the optimal value for this step size will be different to the step size presented in Section 3 since the optimised probability distributions are different.

The NBC algorithm, similar to the NAC algorithm, has good convergence properties. First, the method is guaranteed to converge to the local optimum of the expected cumulative reward function since:

1. the Monte Carlo approximation used in (8) is guaranteed to converge to the true gradient with convergence speed $O(N^{-1/2})$ independent of the number of components in the gradient;
2. the use of *the compatible function approximation* and *the policy gradient iteration with function approximation theorem* in (10) guarantees that the estimate of the "plain" gradient (12) is unbiased and converges to the true gradient; (Sutton et al., 2000)
3. the natural gradient (13) is proven to point in direction of the steepest ascent in a Reimannian space. Consequently, methods using the natural gradient converge to local optimum. (Amari, 1998)

Second, the use of the natural gradient leads to faster converge to the optimal solution when compared to the "plain" gradient methods (Jurčíček et al., 2011b).

To conclude, NBC solves the least squares problem given in (11) to obtain a natural gradient, w, of the expected reward $J(\alpha)$. Finally, the dialogue model parameters are computed as expected values of the prior. The complete NBC algorithm is described in Algorithm 1.

4.2. The dialogue model parameters prior

In order to use NBC in practice, a prior for the model parameters τ is needed. Since the parameters of the BN described in Section 2.2 are parameters of

Algorithm 1 Natural Belief Critic

Let τ be the parameters of the dialogue model

Let $p(\tau; \alpha)$ be a prior for τ parameterised by α

Let N be the number of dialogues sampled in each iteration

Let I be the number of training iterations

Let β be a step size

Input: α_1 - initial parameters of the prior for τ

Input: π - a fixed policy

Output: τ - the updated dialogue model parameters

for $i = 1$ to I **do**

`Collecting statistics:`

  for $n = 1$ to N **do**

    Draw parameters $\tau_n \sim p(\cdot; \alpha_i)$

    Execute the dialogue according to the policy π

    Observe the reward r_n

  end for

  `Critic evaluation:`

  Choose w_i to minimize the sum of the squares of the errors of

$$r_n = \nabla_\alpha \log p(\tau_n; \alpha_i)^T \cdot w_i + C$$

  `Parameter update:`

  $\alpha_{i+1} \leftarrow \alpha_i + \beta w_i$

end for

$\tau = \int p(\tau; \alpha_{I+1}) \tau d\tau$

multiple multinomial distributions, a product of Dirichlet distributions provides a convenient prior.

Formally, for every node $j \in \{1, \ldots, J\}$ in the BN, there are parameters τ_j describing a probability $p(j|par(j); \tau_j)$ where the function $par(j)$ defines the parents of the node j. Let $|par(j)|$ be the number of distinct combinations of values of the parents of j. Then, τ_j is composed of the parameters of $|par(j)|$ multinomial distributions and it is structured as follows: $\tau_j = \left[\tau_{j,1}, \ldots, \tau_{j,|par(j)|}\right]$. Consequently, a prior for τ_j can be formed from a product of Dirichlet distributions: $\prod_{k=1}^{|par(j)|} Dir(\tau_{j,k}; \alpha_{j,k})$, parameterised by $\alpha_{j,k}$. Let the vector $\tau = [\tau_1, \ldots, \tau_J]$ be a vector of all parameters in the BN. Then, the probability $p(\tau; \alpha)$ used in (11) can be defined as:

$$p(\tau; \alpha) = \prod_{j=1}^{J} \prod_{k=1}^{|par(j)|} Dir(\tau_{j,k}; \alpha_{j,k}). \tag{14}$$

which has a closed form log-derivative with respect to α and can be used in (11) to compute the natural gradient.

5. Joint learning of the dialogue model and the policy parameters

5.1. Interleaved training using NAC and NBC algorithms

When a dialogue system uses a stochastic policy and both the model and policy parameters have to be learned, then ideally both the model and policy parameters should be trained jointly. Originally, Jurčíček et al. (2010) suggested interleaved training using the NAC and NBC algorithms. Jurčíček et al. (2010) reported a statistically significant improvement in a simulated evaluation when the dialogue model was initialised with a reasonable set of handcrafted model parameters. The interleaved training scheme was implemented as follows:

- First, the policy parameters were trained using the NAC algorithm while the model parameters were fixed.
- Second, the model parameters were trained using the NBC algorithm while the policy parameters were fixed.

These steps are repeated until the parameters converge.

The difficulty with this interleaving training approach is that neither the policy nor the model parameters can be improved when both are initialised by uninformative parameters. If the model is initialised by uninformative parameters then belief monitoring always computes a belief state which is equivalent to a uniform distribution over all dialogue states regardless of the past observations and system actions. Consequently, the policy learning algorithm cannot improve the policy parameters. If the interleaved algorithm started first with the dialogue model learning then a similar problem would occur. For that reason a different approach to joint learning of both the model and the policy parameters is necessary when training from uninformative parameters. The reader is referred to Jurčíček et al. (2010) for full details and evaluation of this interleaved training scheme.

5.2. Natural Actor and Belief Critic algorithm

As noted in Section 5.1, the interleaved training approach can be used only when an initial dialogue model is available. However, in Jurčíček et al. (2011b), a more general and mathematically founded approach to joint optimisation was suggested. The main idea is that the policy does not have to be fixed as it is in the NBC algorithm and that the NBC and NAC algorithms can be naturally combined to form a joint optimisation method. This generalisation leads to a technique called the Natural Actor and Belief Critic (NABC) algorithm.

The NABC algorithm learns both the parameters α of the prior distribution for the model parameters and the policy parameters θ at the same time. As in NBC, the model parameters τ are sampled at the beginning of each dialogue. However, the objective function function J depends on both θ and α and is expressed as $J(\theta, \alpha) = \int p(H; \theta, \alpha) R(H) dH$ and its gradient can be estimate by:

$$\nabla J(\theta, \alpha) = \frac{1}{N} \sum_{n=1}^{N} \nabla \log p(H_n; \theta, \alpha)(R(H_n) - B), \tag{15}$$

where the probability of the trajectory H is defined as

$$p(H;\theta,\alpha) = p(s_0)p(\tau;\alpha)\prod_{t=1}^{T} p(o_t|s_t)p(s_t|a_{t-1},s_{t-1})\pi(a_{t-1}|b(\cdot|h_{t-1};\tau);\theta).$$

Consequently, the log-gradient $p(H;\theta,\alpha)$ has the following form:

$$\nabla \log p(H;\alpha,\theta) = \left[\nabla_\alpha \log p(\tau;\alpha)^T, \sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|b(\cdot|h_t;\tau);\theta)^T\right]^T.$$

Similar to the derivation of the NBC algorithm, this leads to solving the following set of equations using the least squares method:

$$r_n = \left[\nabla_\alpha \log p(\tau_n;\alpha)^T, \sum_{t=0}^{T_n-1} \nabla_\theta \log \pi(a_t^n|b(\cdot|h_t^n;\tau);\theta)^T\right] \cdot [w_\alpha^T, w_\theta^T]^T + C \quad (16)$$

$$\forall n \in \{1,\ldots,N\}$$

where r_n is the reward observed at the end of each dialogue.

Similar to the NBC algorithm, the solution $[w_\alpha^T, w_\theta^T]^T$ is the natural gradient which can be used to iteratively improve the policy parameters and the prior of the dialogue model parameters: $\theta' \leftarrow \theta + \beta_\theta w_\theta$, $\alpha' \leftarrow \alpha + \beta_\alpha w_\alpha$ and this update converges to a local optimum of the expected cumulative function. The complete NABC algorithm is described in Algorithm 2.

6. Evaluation

The evaluation of the NBC and NABC algorithm was performed on the BUDS dialogue system described in Section 2. The goal of the evaluation was to test whether the NBC and NABC algorithm could learn a better set of parameters then a set of carefully handcrafted model parameters which had been refined over time to optimise performance. First, the NBC and NABC algorithm were used to train the dialogue model and the policy parameters with a user simulator. Some evaluation on the user simulator is provided. Second, a user trial was held to evaluate the system performance using the trained parameters with real users.

Algorithm 2 Natural Actor and Belief Critic

Let τ be the parameters of the dialogue model

Let $p(\tau;\alpha)$ be a prior for τ parameterised by α

Let $\pi(a_t|b_t;\theta)$ be a policy parameterised by θ

Let I be number of executed iterations

Let N be a number of episodes sampled in each iteration

Let T be number of turns in a dialogue

Let β_α and β_θ be step sizes

Input: α_1 - initial parameters of the prior for τ

Input: θ_1 - initial parameters of the policy π

Output: θ - the updated policy parameters

Output: τ - the updated dialogue model parameters

for $i = 1$ to I **do**

`Collecting statistics:`

  for $n = 1$ to N **do**

    Draw parameters $\tau_n \sim p(\cdot;\alpha_i)$

    `Execute the episode:`

    for $t = 0$ to $T_n - 1$ **do**

      Draw action $a_t^n \sim \pi(\cdot|b(\cdot|h_t^n);\theta_i)$

      Observe the reward r_t^n

      Observe o_{t+1}^n

      $h_{t+1} \leftarrow h_t \cup \{a_t^n, o_{t+1}^n\}$

    end for

    Record $r_n = \sum_{t=0}^{T_n-1} r_t^n$

  end for

  `Critic evaluation:`

  Choose $[w_{\alpha,i}, w_{\theta,i}]^T$ to minimize the sum of the squares of the errors of

$$r_n = \left[\nabla_\alpha \log p(\tau_n;\alpha_i)^T, \sum_{t=0}^{T_n-1} \nabla_\theta \log \pi(a_t^n|b(\cdot|h_t^n;\tau);\theta_i)^T\right] \cdot [w_{\alpha,i}^T, w_{\theta,i}^T]^T + C \quad \forall n \in \{1,\ldots,N\}$$

  `Parameter update:`

  $\alpha_{i+1} \leftarrow \alpha_{i+1} + \beta_\alpha w_{\alpha,i}$

  $\theta_{i+1} \leftarrow \theta_{i+1} + \beta_\theta w_{\theta,i}$

end for

$\theta \leftarrow \theta_{I+1}$

$\tau \leftarrow \int p(\tau;\alpha_{I+1})\tau d\tau$

6.1. Training of the dialogue system parameters

For the purpose of the evaluation, four systems were developed:

1. **HDC** is a system using a set of finely tuned handcrafted model parameters and the handcrafted policy described in Section 2.3.3.
2. **NBC** is a system using a set of model parameters estimated by the NBC algorithm and using the handcrafted policy.
3. **NAC** is a system using a set of handcrafted model parameters and a stochastic policy trained by the NAC algorithm.
4. **NABC** is a system using a set of model parameters and policy parameters trained by the NABC algorithm.

Of particular interest is the comparison between the NBC and the HDC system, and the NABC system and the NAC system.

The systems were trained using an agenda-based user simulator (Schatzmann et al., 2005) developed for the Cambridge Restaurant Information domain described in Section 2.2.1. In this domain, users can obtain information about venues such as restaurants, pubs, and coffee shops. The users may provide information about what kind of venue they are looking for, for example the area, the price range, or type of food they prefer. In the case of pubs, the user can also specify whether the venue should have Internet or TV, and whether children should be allowed at the venue. Once a suitable venue has been offered by the system, the user may request additional information such as the address, phone number, postcode, or the prices at that venue.

The user simulator incorporates a semantic concept confusion model, which enables the systems to be trained and tested across a range of semantic error rates. The basic idea of the confusion model is that the correct dialogue act is randomly placed in the N-best list given the error rate used for evaluation - the higher error rate the lower in the N-best list it appears. Then, the rest of the N-best list is filled with automatically generated incorrect hypotheses.

The reward function used for training and evaluation awards -1 in each dialogue turn and at the end of a dialogue it awards 20 for a successful dialogue

and 0 for an unsuccessful one. A dialogue is considered successful if a suitable venue is offered and all further pieces of information are given. In the case where no venue matches the constraints, the dialogue is deemed successful if the system tells the user that no venue matches and a suitable alternative is offered. Since a typical dialogue will require around five or six turns to complete, this implies that around 15 represents an upper bound on the achievable mean reward. As the evaluated algorithms are episodic, the reward is only awarded at the end of a dialogue.

6.1.1. Learning the dialogue model parameters

To train the NBC system, the dialogue model parameters were estimated by the NBC algorithm. The user simulator was set to produce dialogues at error rates on randomly selected by sampling from the uniform distribution over the interval $[0\%, 50\%]$. In each iteration, 16k dialogues were sampled. The prior of the dialogue model was initialised by uninformative (uniform) parameters. There were 577 dialogue model parameters to be estimated.

To assess the ability of the NBC algorithm to optimise the model parameters, the trained NBC system was compared to the HDC system for which no parameter training was necessary. Both the HDC and NBC systems were evaluated over error rates ranging from 0% to 50%. At each error rate, 5000 dialogues were simulated and to reduce the variance of results, this training and evaluation procedure was executed 5 times. The averaged results along with 95% confidence intervals are depicted in Figure 2. The results show that the system with trained BN parameters significantly outperforms the system with handcrafted parameters especially at high error rates. For example, at 35% error rate, the mean reward was increased by 7.34 absolutely from -1.72 to 5.62. Inspection of the results suggests that this improvement can be mostly attributed to the sub-optimality of the handcrafted policy and the ability of the learnt BN parameters to compensate for this. The learning curve for the NBC system is depicted in Figure 3. The figure also shows the performance of the HDC system. One can see that the NBC system outperforms the HDC system

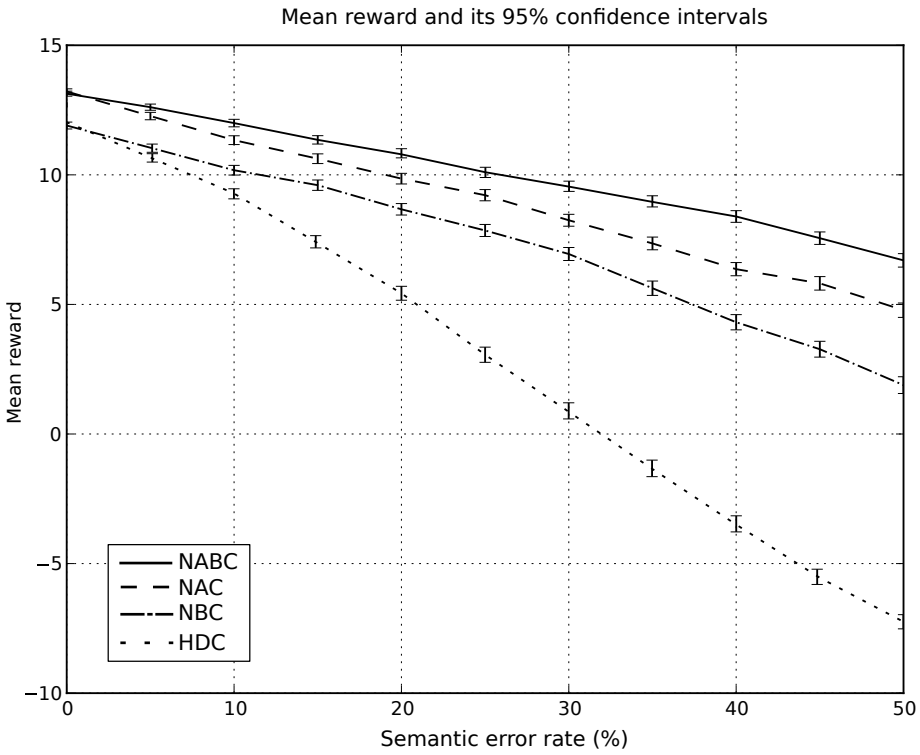

Figure 2: Simulated results for the NBC and NABC algorithms. First, it compares the mean rewards of the handcrafted BN model parameters (HDC) and the model parameters learnt by the NBC algorithm (NBC) when using the fixed handcrafted policy. Second, it compares the mean rewards of the handcrafted BN model parameters (NAC) and the model parameters learnt by the NABC algorithm (NABC) when using the trained policy.

approximately after 24 iterations and after 80 iteration the learning curve for the NBC system levels off. The plotted mean reward is the average mean reward over the error rates 0% to 50% used during training.

6.1.2. Learning the dialogue model and the policy parameters

To train the NABC system, the NABC algorithm was used to jointly estimate a set of dialogue model and policy parameters by running the algorithm for 120 iterations. The setup of the experiment was similar to the previous experiment in Section 6.1.1. The NABC algorithm was executed for 120 iterations

with the simulator set to produce dialogues with error rates between 0% and 50%. The error rates were again uniformly distributed among the dialogues. The total number of sampled dialogues per iteration was 32k. Both the policy parameters and the parameters of the prior of the dialogue model were initialised by uninformative (uniform) parameters. In total, there were 1458 parameters to be estimated.

The performance of the NABC system is compared to the NAC system with the set of handcrafted model parameters and a stochastic policy trained by the NAC algorithm. In the NAC system, only the policy was initialised by uninformative parameters as the dialogue model used the handcrafted parameters. Both the NAC and NABC systems were evaluated over error rates ranging from 0% to 50%. At each error rate, 5000 dialogues were simulated and to reduce the variance of results, this training and evaluation procedure was executed 5 times. The averaged results along with 95% confidence intervals are depicted in Figure 2. The results show that the NABC system significantly outperforms the NAC system. For example, at 35% error rate, the mean reward was increased by 1.68 absolutely from 7.35 to 9.03.

Figure 3 compares the learning curves of the baseline NAC system and the NABC system. In the beginning, the NAC algorithm learns faster as it does not have to learn the model parameters; however, as more iterations of training are completed, the performance of the fully trainable system outperforms the baseline. After 120 iterations, both the model and the policy parameters converge.

6.2. Evaluation with real users

Although the evaluation on a user simulator can provide interesting insight into the performance and behaviour of the tested systems, it is not necessarily a good predictor of the system performance with real human users. Therefore, the baseline handcrafted system and the trained systems, tested in the Section 6.1, were also evaluated in a user trial.

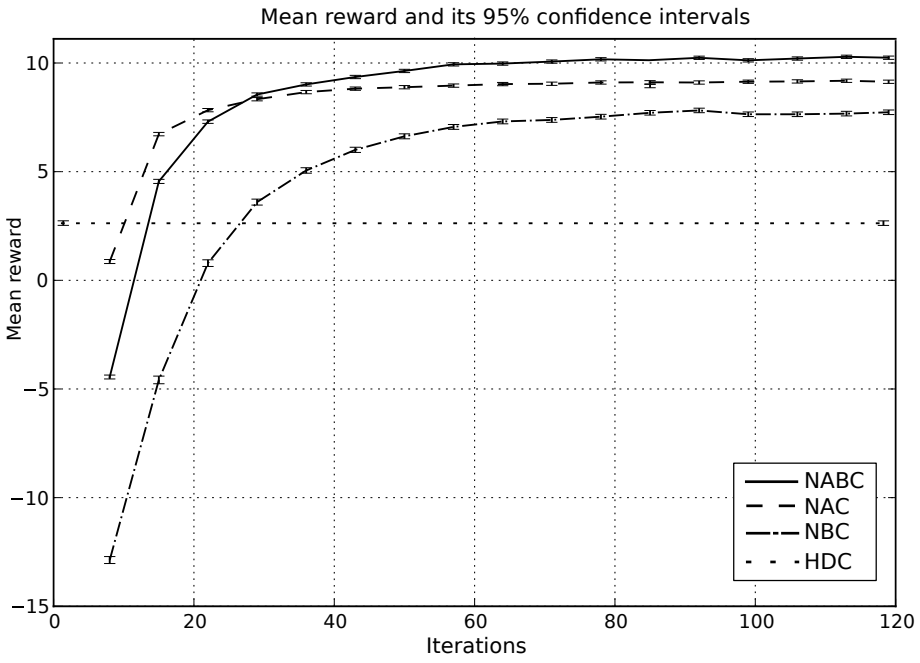

Figure 3: Comparison of the average reward across all error rates during training of the NBC, NAC, and NABC systems. The performance of the handcrafted system (HDC) is plotted as a straight line.

6.2.1. System architecture and trial setup

The BUDS CamInfo dialogue system consists of an ATK-based speech recogniser (Young, 2005), a Phoenix-based semantic parser (Ward, 1991), a POMDP dialogue manager, a template-based natural language generator, and an HMM-based speech synthesiser (Black and Lenzo, 2008). When interacting with the system, the user speech is recognised by the recogniser and an M-best list of user utterance hypotheses is passed to the semantic decoder. The semantic parser outputs an N-best list of dialogue acts typically where $M < N$. This reduction in the number of hypotheses is due to removing duplicate dialogue acts which are produced for similar user utterance hypotheses. For example, if speech recogniser hypotheses differ only in articles then the semantic parser will generate the same dialogue acts.

The N-best list of dialogue acts is used by the dialogue manager to update the belief state. Based on the state and the policy, a system action is produced, again in the form of a dialogue act. The system action is then passed to the natural language generator that converts it to text, which is finally synthesised.

Regarding the trial itself, a telephone-based evaluation framework using Amazon Mechanical Turk (AMT) was used to recruit subjects to evaluate the spoken dialogue systems (Jurčíček et al., 2011a). Users were provided with a web interface containing an introduction, a task description, a telephone number which should be called, and a feedback form. The introduction briefly describes the evaluated dialogue systems, emphasises that the workers should be native English speakers, and provides an example of a typical conversation. The task description presents one randomly selected task from a set of pre-generated tasks. Typically, the tasks describe what kind of venue the user should ask for within the Cambridge Restaurant Information domain (for example "You are looking for a cheap Indian restaurant in the city centre."). For each completed dialogue, the user was asked to provide feedback about that dialogue via a questionnaire. This included questions about the perceived performance of different aspects of the system as well as of the system as a whole. The question "Did you find all the information you were looking for?" aimed to evaluate the overall performance of the dialogue system. To prevent users from submitting feedback without calling any of the systems, the feedback form is only enabled after a unique code has been entered. This code is given to the user by the dialogue system at the end of a call. A routing application was implemented to route an incoming call to the system which was called least times by the user to make sure that a user calls all the systems and the calls are uniformly distributed among the systems. If there were multiple systems with the same number of calls then the target system was randomly selected from these systems.

This setup enabled to collect about 120 calls per day on average to be collected when paying users \$0.20 for each call.

6.2.2. Results

Table 2 provides overall statistics for the corpus of dialogues collected in the trial. To compute Word Error Rate (WER), the recorded audio was transcribed by a set of four selected transcribers recruited on AMT.

The number of calls	2354
The number of turns	25289
The number of users	164
ASR Word error rate	20.1%
Length of the audio	70 hours
Average length of a call	1:47 min

Table 2: Overall statistics for the user trial.

As the number of calls made by each user could not be exactly controlled, the total number of calls per user was analysed to insure that there is no large disparity between the numbers of calls made by different users and that it does not bias the subjective rating. The data from Table 3 shows that users made on average about 14 calls and the median was 9 calls. In other words, 50% of the users made 9 or more calls. When the number of calls is analysed per system then 50% of the users made 3 or more calls to each of the systems. When the corpus was inspected closely, it was found that about 40 users out of 164 did not call all the systems. However, these users made only 65 calls in total which is only a small fraction of the total number of calls. This suggests that the calls from users were well distributed among the systems.

Table 4 presents results obtained during the user trial, comparing the evaluated systems using two metrics: subjective task success and objective task success. The success rate is a proportion of successfully completed dialogues in the trial. In this work, a dialogue is considered to be successful if the dialogue system offers a venue matching the user's constraints and it provides all of the information that the user requested.

A subjective success rate was computed from the users' feedback informa-

System	# users	# calls	average # calls	median of # calls
Total	164	2354	14.35	9
HDC	134	627	4.67	4
NBC	137	573	4.18	3
NAC	140	588	4.20	3
NABC	133	566	4.25	3

Table 3: Number of users and calls per system, average number of calls per user, median of calls per user.

tion, more specifically from their answers to the question "Did you find all the information you were looking for?". However, this metric relies on the ability of the users to accurately rate the performance of the systems. Jurčíček et al. (2011a) analysed the quality of user rated dialogues and observed that all dialogues rated as failed were indeed unsuccessful dialogues; however, many dialogues rated as successful should have been rated as unsuccessful. This suggests that users tend to be overly optimistic in their ratings. For that reason, objective success rates are also computed.

The objective success rate assumes that the users exactly follow the task description. Given the task description that was given to each user, it is possible to determine from the system responses whether the offered venue matched the constraints defined in the goal and whether all required information about the venue was provided. For this purpose, a simple scoring algorithm was implemented. Although this metric solves the problem of false success ratings in the subjective scoring, it underestimates the performance of the evaluated systems since users sometimes divert from the task description, causing some dialogues to be scored negatively as a result. Typically, the user forgets to specify a constraint mentioned in the task description, or forgets to ask for some required additional information about the offered venue, such as the postcode.

Despite these deficiencies, both metrics should provide consistent ratings across the evaluated systems. Only the absolute values have to be interpreted

with some caution as the subjective success rate overestimates and the objective rating underestimates the real performance.

Table 4 presents the subjective and objective success rates from the user trial. The ranking based on the subjective success rates achieved in the trial corresponds to that suggested by the simulations, that is that the NBC system outperforms the HDC system and the NABC system outperforms the NAC system. However, the majority are not statistically significant when tested using a 2-tailed Z-test with the 95% confidence level. Only the differences between the HDC and NAC systems, and the HDC and NABC systems are statistically significant (p-value < 0.001). For the objective success rate ranking, the results suggests that the NBC system outperforms the HDC system; however, the NABC system does not outperform the NAC system. Again the results are not statistically significant.

System	# calls	Subjective Success Rate	Objective Success Rate
HDC	627	82.30% (±2.99)	62.36% (±3.81)
NBC	573	84.47% (±2.97)	63.53% (±3.95)
NAC	588	89.63% (±2.46)	66.84% (±3.79)
NABC	566	90.28% (±2.44)	65.55% (±3.91)

Table 4: Overall results for the user trial. The success rates are followed by their 95% confidence intervals.

Note that in order to achieve statistical significance, a very large number of dialogues would need to be collected. For example, if the Z-test was used to test a difference of 5% in success rate between two systems where the first system has 80% success rate, one would have to collect about 500 calls per system to identify a statistically significant difference at the 95% confidence level. However, if the tested difference was two times smaller, one would need approximately 4 times more dialogues. While collecting a larger number of calls with AMT is possible and easier than in a controlled test environment, it would rapidly become expensive as one call costs on average $0.40 including the

payment to the user, call charges, and transcription of the audio.

There might be several reasons why the difference between the NAC and NABC systems is inconclusive when evaluated on real users. First, as previously noted one would have to collect thousands of calls per system to detect a statistical difference of 1% in success rate. As a result, the observed inconsistency may just be a sampling error. Second, the CamInfo domain is still relatively simple. Therefore, it is not difficult for a trained expert to handcraft and tune a set of reasonable model parameters that are hard to outperform. Third, there is likely to be a mismatch between the simulated user and the real users. As shown, the user simulator and its error simulation are sufficiently accurate to allow a stochastic policy to be trained which significantly outperforms a handcrafted policy. However, the user simulator might not be good enough to allow the model parameters to be estimated sufficiently to outperform the handcrafted parameters when combined with the trained policy. Fourth, the margin for a possible improvement is probably very small. It was observed that the stochastic policy is capable of working well with any "reasonable" model parameters and effectively compensates for sub-optimality of the model parameters. Finally, some of the problems exhibited by the dialogue manager cannot be rectified by a different set of model parameters. For example, the BUDS dialogue manager never confirms information which was acquired with high confidence due to the implementation of slot-level summary actions. Instead, BUDS assumes that it is correct and continues in the dialogue by collecting more information before offering a suitable venue. However, some users are confused when the system does not confirm received information. These users assume that the system did not understand, repeat the already provided information, and fail to answer any further questions. In fact, this is one of the most common reasons why the BUDS dialogue manager fails with real users. Although this particular problem could be fixed by extending the capabilities of the dialogue manager, for instance by defining additional slot-level summary actions, there is currently no parameter in the dialogue model or the policy which could be optimised to deal with this user behaviour.

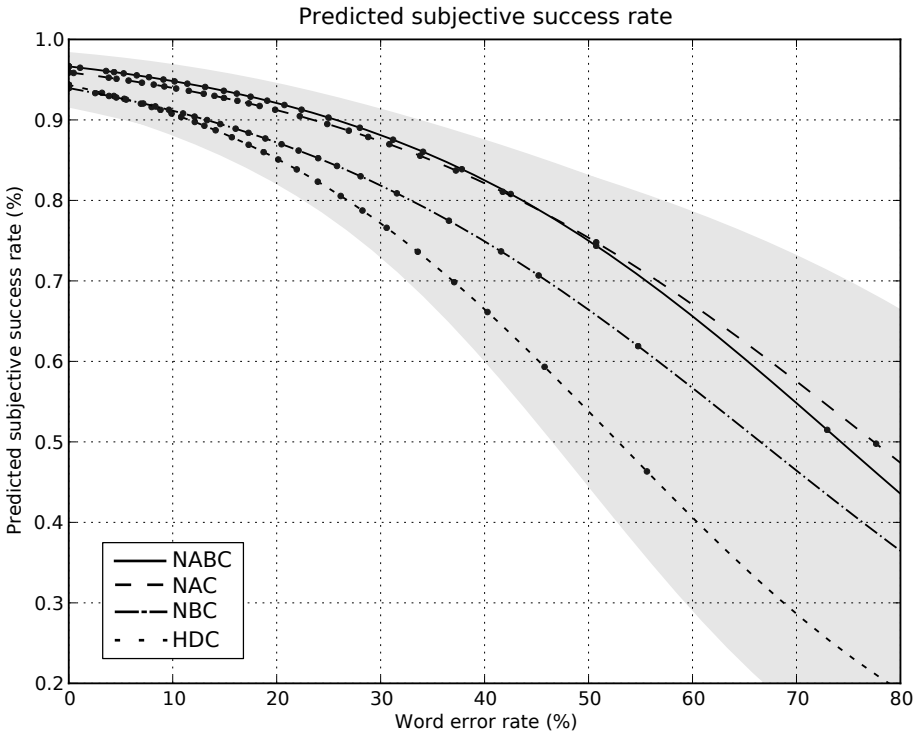

Figure 4: Results for the user trial testing the HDC, NBC, NAC, and NABC systems. The graph plots predicted subjective success rates against the word error rate (WER). Each of the markers represents a group of 20 dialogues at their average word error rate. The grey area depict the 95% confidence intervals for the predicted subjective success rates.

Further insight into the results can be gained by plotting the system performance as a function of the word error rate using logistic regression. Figure 4 shows the dependency between the word error rate measured for each dialogue and the predicted probability of subjective success. Each of the markers represents a group of 20 dialogues at their average word error rate. The grey area depict the 95% confidence intervals for the predicted objective success rates. The figure suggests that the NBC system is more robust to noise when compared to the HDC system as the word error rate increases. However, when comparing the NAC and NABC systems the difference between the systems is not that apparent. At high word error rates (above 50%) the NAC systems ap-

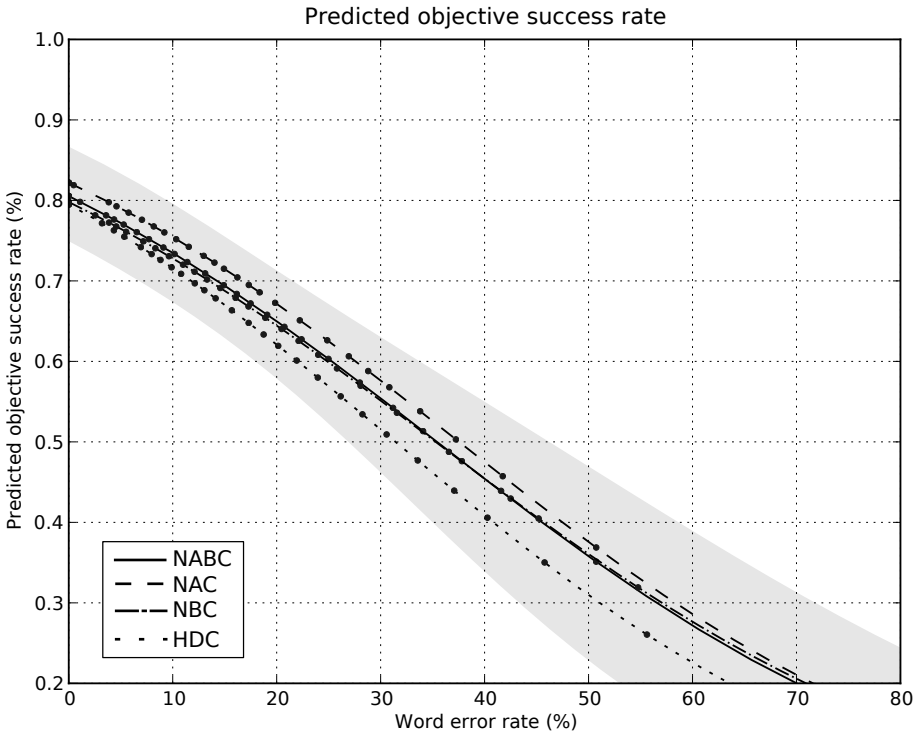

Figure 5: Results for the user trial testing the HDC, NBC, NAC, and NABC systems. The graph plots predicted objective success rates against the word error rate (WER). Each of the markers represents a group of 20 dialogues at their average word error rate. The grey area depict the 95% confidence intervals for the predicted objective success rates.

pears to perform better; however, this is likely to be caused by the small number of collected dialogues at such high word error rates. When looking at word error rates less than 50%, one can see a modest improvement in performance for the NABC system when compared to the NAC system. Figure 5 provides similar results for the objective success rates. Interestingly, the NAC system performs better across all word error rates. However, as noted previously, this difference is not statistically significant. Also, of course, one should keep in mind that the AMT users are still not the "truly real users", as they are paid to do the task. Ultimately, the goal should be to evaluate systems with users who have a genuine information need.

Regarding the absolute values of the objective success rates, one can clearly see that they are about 20% lower when compared to the subjective success rates at the 0% word error rate. As mentioned previously, this is mainly due to the fact that users frequently deviate from the assigned goals. Also, degradation in performance is faster than for the subjective scores. One possible explanation for this is that a higher word error rate makes it harder communicate intelligently with the tested systems. Consequently, users deviate more from the original goal at higher error rates in the attempts to get some information, indeed any information, from the system.

6.2.3. Qualitative and quantitative analysis

Section 6.2.2 provided a set of quantitative measures analysing the efficiency of the evaluated dialogue systems. However, it is also useful to study differences in the behaviour of the systems. Typical qualitative measures characterising behaviour of spoken dialogue systems include (Walker et al., 1997; Dybkjaer et al., 2004):

- number of turns over all and/or for different subtasks,
- how differently the systems react to speech understanding errors,
- recovery time from these errors.

Since the evaluated systems use the same set of summary actions (see Section 2.3.1), the comparative analysis of the behaviour of the systems must concentrate on the differences in the use of these actions. The evaluated dialogue systems use 13 types of summary action in total; however, only few of them significantly affect the quality of the systems. For that reason, only the summary actions which actively manage and decrease uncertainty in the estimate of the dialogue state are analysed. In the CamInfo domain, such summary actions are:

- slot-level summary actions: "request", "confirm", and "select",
- global summary action: "inform about a venue".

Table 5 shows the total number of collected dialogues for each of the evaluated systems, the average number of turns in a dialogue (dialogue length), the average numbers of the "request", "confirm", and "select" slot-level summary actions and the average number of the "inform about a venue" summary action in a dialogue. The statistics show that the optimised dialogue systems achieve shorter dialogues. The exception to this is the NBC system for which the average length of a dialogue is about the same as for the HDC system. When inspecting the use of the summary actions, one can notice the decrease in the average number of slot-level actions for the trained systems. This suggests that the trained systems are more confident about the estimate of the belief state when compared to the handcrafted HDC system and inform about a matching venue earlier. There is no clear difference in the use of the "inform about a venue" summary action among the systems.

	HDC	NBC	NAC	NABC
# dialogues	627	573	588	566
avg. # of turns	10.99	11.01	9.74	9.18
request	3.13	2.65	2.06	1.40
confirm	0.34	0.21	0.03	0.21
select	0.67	0.94	0.16	0.07
inform about a venue	1.86	2.05	2.36	2.13

Table 5: The table shows the number of collected dialogues for each of the evaluated systems, the average number of turns in a dialogue (dialogue length), the average numbers of the "request", "confirm", and "select" slot-level summary actions and the average number of the "inform about a venue" summary action.

To examine differences in the behaviour in detail, it is useful to analyse individual subtasks in a dialogue. In the CamInfo domain, a typical dialogue can be divided into a sequence of subtasks where the first subtask is always *opening* which is then followed by the *initial-offer* subtask. Then, there are several *error-recovery* and/or *request-details* subtasks. Finally, the dialogue ends with

the *closing* subtask. In the *initial-offer* subtasks, a user communicates the venue constraints to the system and it ends by the system informing about a matching venue or that there are no venues matching the inferred constraints. In the *error-recovery* subtask, a user corrects any misunderstanding. Typically, it follows the system offer in which the offered venue does not match exactly the user's constraints. The subtask ends by the system informing about a new matching venue or that there are no venues matching the corrected constraints. In the *request-details* subtask, a user requests details about the last offered venue. A user asks for information such as the address, the post code or the telephone number. The *request-details* subtask usually follows the system offer of a venue which matches the user's constraints.

Table 6 shows statistics for the *initial-offer* subtask. The table presents the number of dialogues with the *initial-offer* subtask for each of the evaluated systems, the average number of turns in the subtask, the average numbers of the "request", "confirm", and "select" slot-level summary actions. Note that the numbers of dialogues with the *initial-offer* subtask differs from the total number of the dialogues. This shows that some of the dialogues ended before the system had an opportunity to offer a venue to the user. Since every *initial-offer* subtask ends with the "inform about a venue" summary action, the average numbers for this summary action are trivially equal to one and for that reason omitted. However, the "inform about a venue" summary action has two subtypes:

1. inform about a venue which matches the inferred constraints,
2. inform that there is no venue that matches the inferred constraints.

Since the tasks used in the user trial were designed to have at least one solution, every time the system informs that there is no matching venue then it signals misunderstanding. Consequently, the ratio of these subtypes characterise the efficiency of the goal inference. As for the overall statistics in Table 5, the results in Table 6 suggest that the trained systems are more confident about the estimated belief state since they use a smaller number of the slot-level summary actions and the average length of the subtask is lower. The ratio between

the subtypes of the "inform about a venue" summary action suggests that the trained systems offer a venue more often where the NABC system achieves the highest proportion of offered venues.

	HDC	NBC	NAC	NABC
# subtasks	582	548	581	554
avg. # of turns	5.18	4.55	3.86	3.64
request	3.30	2.40	1.73	1.35
confirm	0.23	0.13	0.10	0.17
select	0.30	0.27	0.02	0.05
inform about a venue				
- a matching venue	85%	87%	90%	92%
- no matching venue	15%	13%	10%	8%

Table 6: This table shows average number of turns in the initial subtask (subtask length) and average numbers of the "request", "confirm", and "select" slot-level summary actions. The table also shows ratio between two subtypes of the "inform about a venue" summary action.

The subtype (2) of the "inform about a venue" summary action can also be used to automatically identify the *error-recovery* subtasks. As noted above, every time the system informs that there is no matching venue then it signals misunderstanding. It was observed in the data that a user typically continues a dialogue by correcting the system to elicit a valid offer of a venue. As a result, the *error-recovery* subtasks can be defined from the occurrence of the subtype (2) of the "inform about a venue" summary action until the offer of a new venue. Table 7 shows statistics for the *error-recovery* subtask which followed the *initial-offer* subtask. The table shows the number of detected *error-recovery* subtask, the average number of turns in this subtask, the average numbers of the "request", "confirm", and "select" slot-level summary actions and the ratio between the subtypes of the "inform about a venue" summary action. In this case, no tendency in the differences of the length of this subtask is observed. However, one can see that the systems with the trained dialogue model offer a

venue matching the inferred constraints more often than the system with the handcrafted dialogue model parameters. This suggests that the optimisation of dialogue model helps in the error recovery subtask. To verify this hypothesis, the table also shows success rates for the dialogues with the *error-recovery* subtask. The results show that the NBC and NABC systems using an optimised dialogue model are more likely to recover from misunderstanding errors.

	HDC	NBC	NAC	NABC
# subtasks	71	60	56	37
avg. # of turns	1.74	3.93	1.32	1.64
request	0.32	0.56	0.21	0.13
confirm	0.11	0.06	0.00	0.05
select	0.26	2.28	0.05	0.10
inform about a venue				
- a matching venue	24%	68%	41%	89%
- no matching venue	76%	32%	59%	11%
subjective success rate	59.15%	75.00%	67.86%	78.36%

Table 7: This table shows average number of turns in the *error-recovery* subtask (subtask length) and average numbers of the "request", "confirm", and "select" slot-level summary actions. The table also shows ratio between two subtypes of the "inform about a venue" summary action and success rates for the dialogues with the *error-recovery* subtask.

Besides the automatic detection of the *error-recovery* subtasks, a large number of dialogues were manually inspected and labelled for the *error-recovery* subtasks which could not be automatically detected as described above. Unfortunately no clear conclusion could be drawn from the statistics computed from these subtasks.

The *request-details* subtask was also analysed. It was found that the systems exhibited similar behaviour and no significant differences were observed.

7. Discussion

Several techniques for dialogue model learning have been presented in the literature. Recent works include Georgila et al. (2005); Kim et al. (2008); Williams (2008), which used maximum likelihood estimates from a dialogue corpus which was either automatically or manually annotated with dialogue states. Doshi and Roy (2007, 2008); Syed and Williams (2008) developed several methods based on the Expectation-Maximization algorithm; however, it was not shown that these algorithms could be scaled to complex dialogue systems.

Thomson et al. (2010) used Expectation-Propagation (EP) to infer the unobserved dialogue state together with the model parameters. The main advantage of this algorithm is that it is an offline method and it does not rely on annotated data. However, it requires the model to be generative. In other words, the observations must be conditioned on the dialogue state. The belief update (2) was transformed into a generative model using Bayes theorem, and although this form is convenient for using with EM and EP, it is not necessary. Instead, $b(s_t|h_t;\tau)$ can be modelled directly by optimising a target evaluation metric and this is a potentially interesting future direction. Even if the model is generative, it is still useful to directly optimise the evaluation metrics since the structure of a generative model is never perfect due to the many approximations needed. Consequently, additional optimisation of the parameters can improve their performance.

Reinforcement learning techniques have already been used to directly maximise expected reward while learning the policy and model parameters. These techniques usually learn to map observations directly to actions and they use their internal memory[3] to summarise important information from the past observations. For example Wierstra et al. (2010), used recurrent neural networks (RNN), to approximate the policy. This method selects a new system action based on the accumulated information in the internal memory and the last ob-

[3]The internal memory is sometimes called internal state.

servation. Aberdeen (2003) developed several algorithms for controllers with a finite set of internal states, in which the policy maintains beliefs over all of these states. Even though this approach is able to accommodate thousands of states, it does not scale to real word spoken dialogue systems where there are billions of states. Note that if a dialogue system has 10 slots and each slot has 10 different values then there are 10^{10} distinct dialogue states. Recently, Doshi-Velez (2009) described a method based on a non-parametric approach to modelling an infinite POMDP. The described method builds on nonparametric HMMs in which the number of states is not fixed in advance but grows with the available data. This method extends finite state controller algorithms into the infinite domain. Although such methods may seem to be attractive, they are difficult to use in spoken dialogue systems. Since there is no meaning associated with the values in the internal memory, the dialogue system cannot, for example, query a database for requested information.

Another approach to learning a dialogue model is based on model-based reinforcement learning. Recently, Png and Pineau (2011) presented a framework based on a Bayes-Adaptive POMDP algorithm with online approximative planning to learn an observation model. In this work, a dialogue model was factored into a transition model between hidden dialogue states and an observation model, and only learning of the observation model was considered. Also, the presented dialogue model had 25 dialogue states and accepted only 25 observations. Since exact online planning is intractable for even simple tasks, approximative planning based on a tree search algorithm which finds the best action given the current belief state was used. This search must be performed each time the dialogue system has to generate an action. Although pruning techniques can limit the computational cost, the algorithm complexity is still exponential with respect to the depth of the search, the number of the dialogue states and observations. If this is combined with the fact that the belief monitoring becomes also more computationally expensive when the complexity of the dialogue model increases then this approach is currently limited only to small application domains.

Originally, Jurčíček et al. (2010) reported that the NBC algorithm is sensitive to its initialisation and that the initialisation of the prior by uninformative parameters did not lead to convincing results. Consequently, it was suggested that NBC should be used mainly for improving a set of hand crafted parameters or parameters trained by some other method such as maximum likelihood. However, later it was found that the main reason for these difficulties was that the variance of the prior for dialogue model parameters was set too low. For example, the dialogue model priors $Dir(1.0, 1.0, 1.0)$ and $Dir(0.1, 0.1, 0.1)$ are both uninformative; however, they have different variance. After setting the initial prior parameters to 10^{-4} instead 1.0, it was observed that variance in the prior increased enough to achieve successful training from the uninformative parameters. Note that this type of problem does not occur with NAC where the stochastic policy is implemented as a multiclass logistic regression. If all the policy parameters are set to zero then it results in a uniform multinomial distribution over all summary actions with the highest possible variance.

The NBC and NABC algorithms are centred around a least squares regression. In Section 6, the NBC algorithm is used to estimate 565 model parameters and the NABC algorithm estimates 893 policy and 565 model parameters. However, ordinary least squares regression becomes unstable when an approximation with hundreds of parameters has to be estimated. It was observed that a simple ridge regression with the ridge parameter $\lambda = 0.02$ seems to be sufficient to deal with this problem (Hoerl and Kennard, 1970). The complexity of the NBC and the NABC algorithms is mainly defined by the number of the estimated parameters and the number of dialogues sampled in each iteration. Since the core of the algorithms is a least squares method, the algorithms have cubic complexity in the number of parameters estimated and linear complexity in the number of dialogues sampled. Although the complexity of the algorithms is cubic in the number of parameters, the main contributor to the total running time is the number of sampled dialogues. The time necessary to solve the least squares problem in the experiments described in this article took less than 10 seconds (1 processor/core); however, the time needed to sample all dialogues for one

iteration is about 15 minutes on a grid computer with 128 processors/cores. As analysed in Jurčíček et al. (2011b), the use of the natural gradient appears to be necessary. It was observed that techniques based on "plain" gradients, converge significantly slower than methods using "natural" gradients.

The methods presented in this article can be extended in principle to optimise any parameter in the entire dialogue system. The trick is that the learning of non-differentiable parameters of the cumulative reward is replaced by learning differentiable parameters of their prior. When designing such a prior, one should try to capture all important correlations among the model parameters.

8. Conclusion

This article has proposed two novel methods called the Natural Belief Critic (NBC) algorithm and the Natural Actor Belief Critic (NABC) algorithm for estimating the parameters of a POMDP-based dialogue system so as to maximise the expected cumulative reward. The NBC algorithm estimates the model parameters assuming that the policy is fixed whereas the NABC algorithm jointly estimates the dialogue model and the policy parameters. Based on observed rewards obtained in a set of training dialogues, the algorithms estimate the natural gradient of the expected reward of a dialogue system and then adapt the Dirichlet prior distributions of the model parameters and the policy parameters.

Simulations show that both algorithms significantly outperform their respective baselines. The NBC system with the trained dialogue model significantly performed better than the system in which both the model and the policy were handcrafted. The system trained by the NABC algorithm significantly outperforms the handcrafted model parameters and the NAC trained policy. When the systems are evaluated by real users, the results suggest the NBC algorithm again increases the robustness of the system. When evaluating the performance of the system trained by the NABC algorithm, the results are inconclusive. Presumably this is caused by the ability of the stochastic policy to compensate for the suboptimal model parameters on this task and the fact that the model pa-

rameters are still relatively easy to handcraft by an expert. The benefits of the NABC algorithm are likely to become more apparent when used with dialogue systems for more complex domains.

9. Acknowledgement

This research was funded by the EU FP7 Programme under grant agreement 216594 (CLASSIC project: www.classic-project.org). The authors would like to thank Simon Keizer, for his work on the user simulator used here, as well as Milica Gašić, François Mairesse, Kai Yu, and Jorge Prombonas for their useful comments and discussions.

Aberdeen, D. A., 2003. Policy-gradient algorithms for partially observable Markov decision processes. Ph.D. thesis, Australian National University.

Amari, S., 1998. Natural gradient works efficiently in learning. Neural Computation 10 (2), 251–276.

Bishop, C., 2006. Pattern Recognition and Machine Learning. Springer.

Black, A., Lenzo, K., 2008. Flite: a small, fast speech synthesis engine. URL `http://www.speech.cs.cmu.edu/flite/index.html`

Doshi, F., Roy, N., 2007. Efficient model learning for dialog management. In: HRI '07: Proceedings of the ACM/IEEE international conference on Human-robot interaction. ACM, New York, NY, USA, pp. 65–72.

Doshi, F., Roy, N., 2008. Spoken language interaction with model uncertainty: an adaptive human robot interaction system. Connection Science 20 (4), 299–318.

Doshi-Velez, F., 2009. The infinite partially observable markov decision process. In: Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C. K. I., Culotta, A. (Eds.), Advances in Neural Information Processing Systems 22. pp. 477–485.

Dybkjaer, L., Bernsen, N. O., Minker, W., 2004. Evaluation and usability of multimodal spoken language dialogue systems. Speech Communication 43 (1-2), 33 – 54.

Georgila, K., Lemon, O., Henderson, J., 2005. Automatic annotation of COMMUNICATOR dialogue data for learnin dialogue strategies and user simulations. In: In Ninth Workshop on the Semantics and Pragmatics of Dialogue.

Hoerl, A. E., Kennard, R. W., 1970. Ridge Regression: Biased Estimation for Nonorthogonal Problems. Technometrics 12, 55–67.

Jurčíček, F., Keizer, S., Gašić, M., Mairesse, F., Thomson, B., Yu, K., Young, S., 2011a. Real user evaluation of spoken dialogue systems using Amazon Mechanical Turk. In: Proc. Interspeech.

Jurčíček, F., Thomson, B., Keizer, S., Mairesse, F., Gašić, M., Yu, K., Young, S., 2010. Natural Belief-Critic: a reinforcement algorithm for parameter estimation in statistical spoken dialogue systems. In: Kobayashi, T., Hirose, K., Nakamura, S. (Eds.), Proc. Interspeech. ISCA, pp. 90–93.

Jurčíček, F., Thomson, B., Young, S., June 2011b. Natural actor and belief critic: Reinforcement algorithm for learning parameters of dialogue systems modelled as pomdps. ACM Trans. Speech Lang. Process. 7, 6:1–6:26.

Kim, D., Sim, H. S., Kim, K., Kim, J. H., Kim, H., Sung, J. W., 2008. Effects of User Modeling on POMDP-based Dialogue Systems. In: Proceedings of Interspeech.

Konda, V., Tsitsiklis, J., 2000. Actor-critic algorithms. Advances in Neural Information Processing Systems 12.

Peters, J., Vijayakumar, S., Schaal, S., 2005. Natural Actor-Critic. In: European Conference on Machine Learning (ECML). Springer, pp. 280–291.

Png, S., Pineau, J., 2011. Bayesian reinforcement learning for pomdp-based dialogue systems. In: ICASSP '11: International Conference on Acoustics, Speech and Signal Processing.

Roy, N., Pineau, J., Thrun, S., 2000. Spoken dialogue management using probabilistic reasoning. In: ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics, Morristown, NJ, USA, pp. 93–100.

Schatzmann, J., 2008. Statistical user modeling for dialogue systems. Ph.D. thesis, University of Cambridge.

Schatzmann, J., Stuttle, M. N., Weilhammer, K., Young, S., 2005. Effects of the user model on simulation-based learning of dialogue strategies. In: IEEE ASRU '05: Proc. IEEE Workshop Automatic Speech Recognition and Understanding.

Sutton, R., Barto, A., 1998. Reinforcement Learning: An Introduction. Adaptive Computation and Machine Learning. MIT Press, Cambridge, Mass.

Sutton, R., McAllester, D., Singh, S., Mansour, Y., 2000. Policy gradient methods for reinforcement learning with function approximation. In: Advances in Neural Information Processing Systems 12. MIT Press, pp. 1057–1063.

Syed, U., Williams, J. D., 2008. Using automatically transcribed dialogs to learn user models in a spoken dialog system. In: HLT. Morristown, USA, pp. 121–124.

Thomson, B., 2010. Statistical methods for spoken dialogue management. Ph.D. thesis, University of Cambridge.

Thomson, B., Gašić, M., Keizer, S., Mairesse, F., Schatzmann, J., Yu, K., Young, S., 2008a. User study of the Bayesian Update of Dialogue State approach to dialogue management. In: Interspeech 2008. Brisbane, Australia.

Thomson, B., Jurčíček, F., Gašić, M., Keizer, S., Mairesse, F., Yu, K., Young, S., 2010. Parameter learning for POMDP spoken dialogue models. In: IEEE SLT '10: Spoken Language Technology Workshop. pp. 271–276.

Thomson, B., Schatzmann, J., Young, S., 2008b. Bayesian Update of Dialogue State for Robust Dialogue Systems. In: Int Conf Acoustics Speech and Signal Processing ICASSP. Las Vegas.

Thomson, B., Young, S., 2010. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. Computer Speech and Language 24 (4), 562–588.

Walker, M. A., Litman, D. J., Kamm, C. A., Abella, A., 1997. Evaluating interactive dialogue systems: extending component evaluation to integrated system evaluation. In: Interactive Spoken Dialog Systems on Bringing Speech and NLP Together in Real Applications. ISDS '97. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 1–8.

Ward, W., 1991. Understanding Spontaneous Speech. In: Proc Int Conf Acoustics, Speech and Signal Processing. Toronto, Canada, pp. 365–368.

Wierstra, D., Förster, A., Peters, J., Schmidhuber, J., 2010. Recurrent policy gradients. Logic Journal of the IGPL 18 (5), 620–634.

Williams, J. D., 2008. Integrating expert knowledge into POMDP optimization for spoken dialog systems. In: Proceedings of the AAAI-08 Workshop on Advancements in POMDP Solvers.

Williams, J. D., Young, S., November 2005. Scaling Up POMDPs for Dialog Management: The Summary POMDP Method. In: IEEE ASRU '05: Proc. IEEE Workshop Automatic Speech Recognition and Understanding. Cancun, Mexico.

Williams, J. D., Young, S., 2007a. Partially Observable Markov Decision Processes for Spoken Dialog Systems. Computer Speech and Language 21 (2), 393–422.

Williams, J. D., Young, S., 2007b. Scaling POMDPs for Spoken Dialog Management. IEEE Audio, Speech and Language Processing 15 (7), 2116–2129.

Williams, R. J., 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning 8.

Young, S., 2002. Talking to Machines (Statistically Speaking). In: Int. Conf. Spoken Language Processing. Denver, Colorado.

Young, S., 2005. ATK: An Application Toolkit for HTK.
URL `http://mi.eng.cam.ac.uk/research/dialogue/atk_home`

Young, S., 2007. CUED standard dialogue acts: `http://mi.eng.cam.ac.uk/research/dialogue/LocalDocs/dastd.pdf`. Tech. rep., Cambridge University Engineering Dept.

Young, S., Gašić, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., Yu, K., 2010. The Hidden Information State Model: a practical framework for POMDP-based spoken dialogue management. Computer Speech and Language 24 (2), 150–174.