Hybrid Reinforcement/Supervised Learning of Dialogue Policies from Fixed Data Sets

James Henderson*
University of Geneva

Oliver Lemon**
University of Edinburgh

Kallirroi Georgila**
University of Edinburgh

We propose a method for learning dialogue management policies from a fixed data set. The method addresses the challenges posed by Information State Update (ISU)-based dialogue systems, which represent the state of a dialogue as a large set of features, resulting in a very large state space and a huge policy space. To address the problem that any fixed data set will only provide information about small portions of these state and policy spaces, we propose a hybrid model that combines reinforcement learning with supervised learning. The reinforcement learning is used to optimize a measure of dialogue reward, while the supervised learning is used to restrict the learned policy to the portions of these spaces for which we have data. We also use linear function approximation to address the need to generalize from a fixed amount of data to large state spaces. To demonstrate the effectiveness of this method on this challenging task, we trained this model on the COMMUNICATOR *corpus, to which we have added annotations for user actions and Information States. When tested with a user simulation trained on a different part of the same data set, our hybrid model outperforms a pure supervised learning model and a pure reinforcement learning model. It also outperforms the hand-crafted systems on the* COMMUNICATOR *data, according to automatic evaluation measures, improving over the average* COMMUNICATOR *system policy by 10%. The proposed method will improve techniques for bootstrapping and automatic optimization of dialogue management policies from limited initial data sets.*

1. Introduction

In the practical development of dialogue systems it is often the case that an initial corpus of task-oriented dialogues is collected, either using "Wizard of Oz" methods or a prototype system deployment. This data is usually used to motivate and inspire a new hand-built dialogue system or to modify an existing one. However, given the

* Université de Genève, Département d'Informatique, Battelle-bâtiment A, 7 route de Drize, 1227 Carouge, Switzerland. E-mail: james.henderson@cui.unige.ch.
** University of Edinburgh, 2 Buccleuch Place, Edinburgh EH8 9LW, UK. E-mail: {olemon, kgeorgil}@inf.ed.ac.uk.

Submission received: 18 November 2005; revised submission received: 18 October 2006; accepted for publication: 21 September 2007.

existence of such data, it should be possible to exploit machine learning methods to automatically build and optimize a new dialogue system. This objective poses two questions: what machine learning methods are effective for this problem? and how can we encode the task in a way which is appropriate for these methods? For the latter challenge, we exploit the Information State Update (ISU) approach to dialogue systems (Bohlin et al. 1999; Larsson and Traum 2000), which provides the kind of rich and flexible feature-based representations of context that are used with many recent machine learning methods, including the linear function approximation method we use here. For the former challenge, we propose a novel hybrid method that combines reinforcement learning (RL) with supervised learning (SL).

The focus of this article is to establish effective methods for using fixed corpora of dialogues to automatically optimize complex dialogue systems. To avoid the need for extensive hand-crafting, we allow rich representations of context that include all the features that might be relevant to dialogue management decisions, and we allow a broad set of dialogue management decisions with very few constraints on when a decision is applicable. This flexibility simplifies system design, but it leads to a huge space of possible dialogue management policies, which poses severe difficulties for existing approaches to machine learning for dialogue systems (see Section 1.1). Our proposed method addresses these difficulties without the use of user simulations, feature engineering, or further data collections.

We demonstrate the effectiveness of the proposed method on the COMMUNICATOR corpora of flight-booking dialogues. Our method ("hybrid learning" with linear function approximation) can learn dialogue strategies that are better than those learned by standard learning methods, and that are better than the (in this case hand-coded) strategies present in the original corpora, according to a variety of metrics. To evaluate learned strategies we run them with simulated users that are also trained on (different parts of) the COMMUNICATOR corpora, and automatically score the simulated dialogues based on how many information "slots" they manage to collect from users ("filled slots"), whether those slots were confirmed ("confirmed slots"), and how many dialogue turns were required to do so. Later work has shown these metrics to correlate strongly with task completion for real users of the different policies (Lemon, Georgila, and Henderson 2006).

The main contributions of the work are therefore in empirically demonstrating that:

- limited initial data sets can be used to train complex dialogue policies, using a novel combination of supervised and reinforcement learning; and
- large, feature-based representations of dialogue context can be used in tractable learning of dialogue policies, using linear function approximation.

In this article, after a discussion of related work, we outline the annotations we have added to the COMMUNICATOR data, then present the proposed learning method, and describe our evaluation method. Finally, we present the evaluation results and discuss their implications.

1.1 Related Work

As in previous work on learning for dialogue systems, in this article we focus on learning dialogue management policies. Formally, a dialogue management policy is a

mapping from a dialogue context (a.k.a. a state) to an action that the system should take in that context. Because most previous work on dialogue systems has been done in the context of hand-crafted systems, we use representations of the dialogue context and the action set based on previous work on hand-crafted dialogue systems. Our main novel contribution is in the area of learning, where we build on previous work on automatically learning dialogue management policies, discussed subsequently.

The ISU approach to dialogue (Bohlin et al. 1999; Larsson and Traum 2000) employs rich representations of dialogue context for flexible dialogue management. **Information States** are feature structures intended to record all the information about the preceding portion of the dialogue that is relevant to making dialogue management decisions. An example of some of the types of information recorded in our Information States is shown in Figure 1, including filled slots, confirmed slots, and previous speech acts. Previous work has raised the question of whether dialogue management policies can be learned (Levin and Pieraccini 1997) for systems that have only a limited view of the dialogue context, for example, not including prior speech act history (see the following).

One prominent representation of the set of possible system actions is the DATE scheme (Walker and Passonneau 2001). In particular, this representation is used in the COMMUNICATOR corpus annotation (Walker, Passonneau, and Boland 2001), discussed herein. The DATE scheme classifies system actions in terms of their Conversational Domain, Speech Act, and Task. For example, one possible system action is ⟨`about_task`,

StateNumber:13

DIALOGUE LEVEL
Turn: user
Speaker: user
ConvDomain: [about_task]
SpeechAct: [provide_info]
AsrInput: (date_time) october three first late morning (date_time)
TransInput: (date_time) october thirty first late morning (date_time)

TASK LEVEL
Task: [depart_time]
FilledSlot: [depart_time]
FilledSlotValue: [late morning]
ConfirmedSlot: [dest_city]

LOW LEVEL
WordErrorRate: 20.00

HISTORY LEVEL
FilledSlotsStatus: [orig_city], [dest_city], [depart_time]
FilledSlotsValuesStatus: [hartford connecticut], [orlando florida], [late morning]
ConfirmedSlotsStatus: [orig_city], [dest_city]
SpeechActsHist: [yes_answer], opening_closing, [ ], opening_closing, instruction, request_info,
[provide_info], implicit_confirm, request_info, [provide_info], implicit_confirm,
request_info, [provide_info]
TasksHist: [null], meta_greeting_goodbye, [ ], meta_greeting_goodbye, meta_instruct, orig_city, [orig_city],
orig_city, dest_city, [dest_city], dest_city, depart_arrive_date, [depart_time]
FilledSlotsHist: [null], [ ], [orig_city], [dest_city], [depart_time]
FilledSlotsValuesHist: [yes], [ ], [hartford connecticut], [orlando florida], [late morning]
ConfirmedSlotsHist: [ ], [ ], [ ], [orig_city], [dest_city]

Figure 1
Example fields from an Information State annotation. User-provided information is in square brackets.

request_info, dest_city⟩, which corresponds to a system utterance such as *What is your destination city?* The specific instantiation of this scheme, and our extensions to it, are discussed in Section 1.2.

Machine-learning approaches to dialogue management attempt to learn optimal dialogue policies from corpora of simulated or real dialogues, or by generating such data during automatic trial-and-error exploration of possible policies. Automatic optimization is desirable because of the high cost of developing and maintaining hand-coded dialogue managers, and because there is no guarantee that hand-coded dialogue management strategies are good. Several research groups have developed reinforcement learning approaches to dialogue management, starting with Levin and Pieraccini (1997) and Walker, Fromer, and Narayanan (1998). Previous work has been restricted to limited dialogue context representations and limited sets of actions to choose among (Walker, Fromer, and Narayanan 1998; Goddeau and Pineau 2000; Levin, Pieraccini, and Eckert 2000; Roy, Pineau, and Thrun 2000; Scheffler and Young 2002; Singh et al. 2002; Williams and Young 2005; Williams, Poupart, and Young 2005a).

Much of the prior work in RL for dialogue management focuses on the problem of choosing among a particular limited set of actions (e.g., confirm, don't confirm) in specific problematic states (see, e.g., Singh et al. 2000a). This approach augments, rather than replaces, hand-crafted dialogue systems, because the vast majority of decisions, which are not learned, need to be specified by hand. In contrast, we tackle the problem of learning to choose among any possible dialogue actions for almost every possible state.

In addition, all prior work has used only a limited representation of the dialogue context, often consisting only of the states of information slots (e.g., destination city filled with high confidence) in the application (Goddeau and Pineau 2000; Levin, Pieraccini, and Eckert 2000; Singh et al. 2000a, 2000b, 2002; Young 2000; Scheffler and Young 2002; Williams, Poupart, and Young 2005a, 2005b; Williams and Young 2005; Pietquin and Dutoit 2006b), with perhaps some additional low-level information (such as acoustic features [Pietquin 2004]). Only recently have researchers experimented with using enriched representations of dialogue context (Gabsdil and Lemon 2004; Lemon et al. 2005; Frampton and Lemon 2006; Rieser and Lemon 2006c), as we do in this article. From this work it is known that adding context features leads to better dialogue strategies, compared to, for example, simply using the status of filled or confirmed information slots as has been studied in all prior work (Frampton and Lemon 2006). In this article we explore methods for scalable, tractable learning when using all the available context features.

Reinforcement Learning requires estimating how good different actions will be in different dialogue contexts. Because most previous work has only differentiated between a small number of possible dialogue contexts, they have been able to perform these estimates for each state independently (e.g., Singh et al. 2002; Pietquin 2004). In contrast, we use function approximation to allow generalization to states that were not in the training data. Function approximation was also applied to RL by Denecke, Dohsaka, and Nakano (2005), but they still use a relatively small state space (6 features, 972 possible states). They also only exploit data for the 50 most frequent states, using what is in effect a Gaussian kernel to compute estimates for the remaining states from these 50 states. This is a serious limitation to their method, because a large percentage of the data is likely to be from less frequent states, and thus would be ignored. In our data set, we found that state frequencies followed a Zipfian (i.e., large-tailed) distribution, with 61% of the system turns having states that only occurred once in the data.

Another source of variation between learning approaches is the extent to which they train on data from simulated users of different kinds, rather than train on data gathered from real user interactions (as is done in this article). Simulated users are generally preferred due to the much smaller development effort involved, and the fact that trial-and-error training with humans is tedious for the users. However, the issues of how to construct and then evaluate simulated users are open problems. Clearly there is a dependency between the accuracy of the simulation used for training and the eventual dialogue policy that is learned (Schatzmann et al. 2005). Current research attempts to develop metrics for user simulation that are predictive of the overall quality of the final learned dialogue policy (Schatzmann, Georgila, and Young 2005; Schatzmann et al. 2005; Georgila, Henderson, and Lemon 2005; Georgila, Henderson, and Lemon 2006; Rieser and Lemon 2006a; Schatzmann et al. 2006; Williams 2007). Furthermore, several approaches use simple probabilistic simulations encoded by hand, using intuitions about reasonable user behaviors (e.g., Pietquin 2004; Frampton and Lemon 2005; Pietquin and Dutoit 2006a), whereas other work (e.g., Scheffler and Young 2001, 2002; Georgila, Henderson, and Lemon 2005; Georgila, Henderson, and Lemon 2006; Rieser and Lemon 2006a) builds simulated users from dialogue corpora. We use the latter approach, but only in the evaluation of our learned policies.

No matter which method is chosen for user simulation, a simulated user is still clearly different from a human user. Therefore, it is important to learn as much as possible from the data we have from human users. In addition, the huge policy space makes policy exploration with simulated users intractable, unless we can initialize the system with a good policy and constrain the policy exploration. This also requires learning as much as possible from the initial set of data. Therefore, in this article we investigate using a fixed corpus of dialogues to automatically optimize dialogue systems. No user simulation is involved in training, thus avoiding the issue of dependency on the quality and availability of user simulations.

Previous work on RL has made use of policy exploration (Sutton and Barto 1998), where new data is generated for each policy that is considered during the course of learning (for example using simulated users). Indeed, this is often considered an integral part of RL. In contrast, we choose to learn from a fixed data set, without policy exploration. This is motivated by the fact that real dialogue corpora are very expensive to produce, and it is often not practical to produce new real dialogues during the course of learning. Singh et al. (2002) manage to perform one iteration of policy exploration with real data, but most work on RL requires many thousands of iterations. As discussed previously, this motivates using simulated data for training, but even if accurate dialogues can be automatically generated with simulated users, training on simulated dialogues does not replace the need to fully exploit the real data, and does not solve the sparse data problems that we address here. With a very large state space, it will never be tractable for policy exploration to test a new policy on even a reasonable proportion of the states. Thus we will inevitably need to stop policy exploration with a policy that has not been sufficiently tested. In this sense, we will be in a very similar situation to learning from a fixed data set, where we don't have the option of generating new data for new states. For this reason, the solution we propose for learning from fixed data sets is also useful for learning with policy exploration.

There have been some proposals in RL for learning a policy that is different from that used to generate the data (called "off-policy" learning), but these methods have been found not to work well with linear function approximation (Sutton and Barto 1998). They also do not solve the problem of straying from the region of state space that has been observed in the data, discussed subsequently.

1.2 The COMMUNICATOR Domain and Data Annotation

To empirically evaluate our proposed learning method, we apply it to the COMMUNICATOR domain using the COMMUNICATOR corpora. The COMMUNICATOR corpora (2000 [Walker et al. 2001] and 2001 [Walker et al. 2002b]) consist of human–machine dialogues (approximately 2,300 dialogues in total). The users always try to book a flight, but they may also try to select a hotel or car rental. The dialogues are primarily "slot-filling" dialogues, with some information being presented to the user after the system thinks it has filled (or confirmed) the relevant slots. These corpora have been previously annotated using the DATE scheme, for the Conversational Domain, Speech Act, and Task of each *system* utterance (Walker and Passonneau 2001; Walker, Passonneau, and Boland 2001). In addition, the results of user questionnaires are available, but only for the 2001 corpus.

Table 1 shows some statistics for the two collections. In the 2000 collection each turn contains only one utterance but in the 2001 corpus a turn may contain more than one utterance. More details about the COMMUNICATOR corpora can be found in Walker, Passonneau, and Boland (2001) and Walker et al. (2001, 2002a).

We used a hand-crafted automatic system (Georgila, Lemon, and Henderson 2005; Georgila et al., submitted) to assign Speech Acts and Tasks to the *user* utterances, and to compute state representations for each point in the dialogue (i.e., after every utterance). Although we annotated the whole 2000 and 2001 corpora, because we need the results of user questionnaires (as discussed subsequently), we only make use of the 2001 data for the experiments reported here. The 2001 data has eight systems, 1,683 dialogues, and 125,388 total states, two thirds of which result from system actions and one third from user actions. The annotation system is implemented using DIPPER (Bos et al. 2003) and OAA (Cheyer and Martin 2001), using several OAA agents (see Georgila, Lemon, and Henderson, 2005, and Georgila et al., submitted, for more details). Following the ISU approach, we represented states using Information States, which are feature structures intended to record all the information about the preceding portion of the dialogue that is relevant to making dialogue management decisions. An example of some of the types of information recorded in an Information State is shown in Figure 1, including filled slots, confirmed slots, and previous speech acts.

Given this corpus, we need to learn a dialogue management policy that maps these state representations to effective system actions. As the example in Figure 1 illustrates, there are a large number of features in dialogue states that are potentially relevant to

Table 1
Statistics for the 2000 and 2001 COMMUNICATOR data.

	Year		
	2000	2001	Total
Number of dialogues	648	1683	2331
Number of turns	24,728	78,718	103,446
Number of system turns	13,013	39,419	52,432
Number of user turns	11,715	39,299	51,014
Number of utterances	24,728	89,666	114,394
Number of system utterances	13,013	50,159	63,172
Number of user utterances	11,715	39,507	51,222
Number of system dialogue acts	22,752	85,881	108,633

dialogue management, and thus should not be excluded from the state representations we use in learning. This leads to a very large space of possible states (over 10^{386} states are theoretically possible in our model), with a very high chance that a state encountered in testing will not be exactly the same as any state encountered in training. This fact motivates, if not requires, the use of approximation methods.

The complexity of the COMMUNICATOR domain is also manifested in the large number of system actions that the dialogue management policy needs to choose between. The DATE scheme representation of system actions implies that each possible triple of values for the Conversational Domain, Speech Act, and Task is a different action. In addition, we have added `release_turn` and `end_dialogue` actions. There are a total of 74 system actions that occur in the annotated COMMUNICATOR data.

2. Reinforcement Learning with a Fixed Data Set

We use the annotated COMMUNICATOR data to train a Reinforcement Learning system. In RL, the objective of the system is to maximize the reward it gets during entire dialogues. Rewards are defined to reflect how successful a dialogue was, so by maximizing the total reward the system optimizes the quality of dialogues. The difficulty is that, at any point in the dialogue, the system cannot be sure what will happen in the remainder of the dialogue, and thus cannot be sure what effect its actions will have on the total reward at the end of the dialogue. Thus the system must choose an action based on the average reward it has observed previously after it has performed that action in states similar to the current one. This average is the **expected future reward**.

The core component of any RL system is the estimation of the expected future reward (called the Q-function). Given a state and an action that could be taken in that state, the Q-function tells us what total reward, on average, we can expect between taking that action and the end of the dialogue.[1] Once we have this function, the optimal dialogue management policy reduces to simply choosing the action that maximizes the expected future reward for the current state.

Our proposal for RL with fixed data sets uses two main techniques. The first is the use of function approximation to estimate the expected future reward. We claim that linear function approximation is an effective way to generalize from a limited data set to a large space of state–action pairs. The second technique is a novel hybrid learning method that combines RL with supervised learning (SL). SL is used to characterize how much data we have for each area of the state–action space (also using linear function approximation). Our hybrid policy uses SL to avoid state–action pairs for which we do not have enough data, while using RL to maximize reward within the parts of the space where we do have enough data. We claim that this is an effective solution to the problem of learning complex tasks from fixed data sets.

2.1 Defining Dialogue Reward

To apply RL to the COMMUNICATOR data, we first have to define a mapping $r(d, i)$ from a dialogue d and a position in that dialogue i to a reward value. This reward function is computed using the reward level of annotation in the COMMUNICATOR data, which was

1 The expected future reward also depends on the dialogue management policy that the system will use in the future. This self-referential nature of RL is the topic of much RL research, and will be discussed more in the following.

extracted from user questionnaires and task completion measures. For all states other than the final state, we provide a reward of −1 if the state follows a system action, and 0 otherwise. This encodes the idea that, all other things being equal, short dialogues are better than long ones. For the final state we provide a reward that is the sum of the rewards for each feature in the reward annotation. "Actual Task Completion" and "Perceived Task Completion" are both worth a reward of 100 if they are non-zero (i.e., true), and 0 otherwise. The remaining reward features have values ranging from 1 to 5 in the annotation (where 5 is the best), which we rescale to the range 0 to 1 (1 converts to 0, 5 converts to 1). Their reward is their rescaled value times the weight shown in Table 2. The relative values of these later weights were determined by the empirical analysis reported in Walker et al. (2001) within the PARADISE evaluation framework (Walker, Kamm, and Litman 2000).

2.2 Estimating the Expected Future Reward

Given this definition of reward, we want to find an estimate $Q(s_i, a)$ of the expected future reward, which is the expected value ("$E[\ ]$" in Equation 1) of the total reward between taking action a in state s_i and the end of the dialogue. This expectation is a sum over all possible future dialogues d, weighted by the probability of the dialogue given that we have performed action a in state s_i.

$$Q(s_i, a) \approx E_{d|s_i,a}[\sum_{j>i} r(d, j)] = \sum_{d} (P(d|s_i, a) \sum_{j>i} r(d, j)) \quad (1)$$

Given that the number of possible future dialogues $d = \langle s_{i+1}, \ldots, s_{n_d} \rangle$ is exponential in the length of the sequences, it is not surprising that estimating the expected reward over these sequences can be very difficult.

The ISU framework is significantly different from the frameworks used in previous work on reinforcement learning for dialogue management, in that the rich context representation makes the number of possible states extremely large. Having a large number of states is a more realistic scenario for practical, flexible, and generic dialogue systems, but it also makes many RL approaches intractable. In particular, with a large number of states it is not possible to learn estimates of the expected future reward for every state, unless we can exploit commonalities between different states. The feature-based nature of ISU state representations expresses exactly these commonalities between states through the features that the states share. There are a number of techniques that could be used for RL with feature-based representations of states, but the simplest and most efficient is linear function approximation.

Table 2
The weights used to compute a dialogue's final reward value, multiplied by values between 0 and 1 computed from user responses.

Actual task completion	100
Perceived task completion	100
Task ease	36
Comprehension ease	28
System behaved as expected	32
Future use	36

We use linear function approximation to map from a vector of real valued features $f(s)$ for the state s to a vector of estimates $Q(s,a)$, one estimate for each a. The trained parameters of the linear function are a vector of weights w_a for each action a. Given weights trained on a given data set, an estimate $Q_{data}(s,a)$ of the expected future reward given a state s and an action a is the inner product of the state vector $f(s)$ and the weight vector w_a.[2]

$$Q_{data}(s,a) = f(s)^T w_a = \sum_i f_i(s) w_{ai} \tag{2}$$

This approximation method has the effect of treating two states as similar if they share features. During learning, updating the estimate $Q_{data}(s,a)$ for one observed state s will also update the estimate $Q_{data}(s',a)$ for all other states s' to the extent that s' shares features with s. This updating happens via the weights w_a; if s has feature i then updating the estimate $Q_{data}(s,a)$ will change w_{ai}, which will in turn change $Q_{data}(s',a)$ for any s' that also has feature i. Thus each feature represents a dimension with respect to which two states can be similar or different. This similarity measure is known as a linear kernel.

This is the first time that linear function approximation has been used for learning dialogue strategies. Denecke, Dohsaka, and Nakano (2005) also use function approximation, but there the notion of similarity used during learning is Euclidean distance, rather than shared features. In effect, Denecke, Dohsaka, and Nakano use a Gaussian kernel, whereas we use a linear kernel.

To train the weights of the linear approximation $Q_{data}(s,a)$, we employed a standard RL learning method called SARSA(λ) (Sutton and Barto 1998). This method learns based on two criteria, with a parameter λ used to weight their relative influence. The first criterion comes from temporal-difference learning: the current estimate for the Q-function should (on average) equal the reward from the next state plus the estimate for the expected future reward at the next state. The second criterion comes directly from the observed reward: The current estimate for the Q-function should (on average) equal the reward observed for the remainder of the dialogue. The combination of these two criteria makes learning faster than using either one alone. Gradient descent learning is applied to the weights; at each step of learning, the weights are updated so as to make the Q-function better fit this combined criterion.

Whereas the weights w_a are learned from data, the mapping $f(s)$ from states to vectors must be specified beforehand. Because each value $f_i(s)$ in these vectors represents a possible commonality between states, it is through the definition of $f(s)$ that we control the notion of similarity that will be used by the linear function approximation. The definition of $f(s)$ we are currently using is a straightforward mapping from attribute–value pairs in the Information State s to values in the vector $f(s)$.

The state vector mapping $f(s)$ was computed using the first four levels of our state annotations for the COMMUNICATOR data (i.e., the Dialogue, Task, Low, and History levels shown in Figure 1). The values of the attributes in these annotations were converted to features of three types. For attributes that take numbers as values, we used a simple function to map these numbers to a real number between 0 and 1, with the absence of any value being mapped to 0 (resulting in six features, e.g., `StateNumber`).

2 We will use the notation $x^T y$ to denote the inner product between vectors x and y (i.e., "x transpose times y"). w_{ai} is the ith element of the vector w_a.

For attributes that can have arbitrary text as their values, we used 1 to represent the presence of text and 0 to represent no value (resulting in two features, e.g., `AsrInput`). The remaining attributes all have either a finite set of possible values, or a list of such values.

The vast majority of our features are constructed from this third set of attributes. First, to reflect the importance of speech act–task pairs (which we use to define both system and user actions), we construct a new `SpeechAct-Task` attribute whose value is the concatenation of the values for the `SpeechAct` and `Task` attributes. The same is done for the `SpeechActsHist` and `TasksHist` attributes. Second, attributes with a list value (i.e., the `...Hist` and `...Status` attributes, plus user actions[3]) are converted to a set of attribute–value pairs consisting of the attribute and each value in the list (resulting in 509 features, e.g., `FilledSlotsStatus:[orig_city]`). Note that this conversion loses the ordering between the values in the list. In the case of `SpeechAct`, `Task`, and `SpeechAct-Task` attributes that have list values (which result from turns in which a user performs more than one action), we also include the whole list as a value for the attribute[4] (resulting in 364 features, e.g., `SpeechAct:[no_answer,provide_info]`). Finally, attributes with single values are assigned features (which result in 401 features, e.g., `Speaker:user`).

From this set of potential features, we only use those that occur in the data at least five times.[5] (Only these features are included in the feature counts given previously.) Each feature is assigned an element of the vector $f(s)$ that is 1 if that feature is present in the state and 0 if it is not. In total there are 1,282 features.

One advantage of using linear function approximation is that the learning method can be kept fairly simple, while still incorporating domain knowledge in the design of the mapping to feature vectors. One area of future research is to investigate more complicated mappings to feature vectors $f(s)$. This would involve making use of kernel-based methods. Kernels are used to compensate for the oversimplicity of linear functions, and can be used to express more complicated notions of commonality between states (Shawe-Taylor and Cristianini 2004).

2.3 Pure RL and SL Policies

Given the estimate of the expected future reward $Q_{data}(s,a)$ discussed in the previous section, one obvious approach would use this estimate to define the dialogue policy. This "pure RL" policy simply selects the action a with the highest $Q_{data}(s,a)$ given the state s. As demonstrated by the evaluation in Section 3, this policy performs very badly.

Inspection of the actions chosen by the pure RL policy indicates that this policy is very different from the policy observed in the COMMUNICATOR data; the pure RL policy almost never chose the same action as was in the data. This means that the actions that have been learned to have the best future reward for a state are not the ones that were

3 Because in the 2001 COMMUNICATOR data users may perform more than one action in a single turn, a user's action is potentially a list of speech act–task pairs. These are annotated as lists of speech acts plus lists of tasks, to which we add lists of speech act–task pairs. Histories of these lists (i.e., lists of lists) are first flattened and then treated like other lists.

4 These "list" values are more accurately described as set values, because we do not encode the ordering of the values in the list.

5 We also do not include the `...Value...` attributes, such as `FilledSlotValue`, which specify the actual fillers for slots.

typically chosen by the COMMUNICATOR systems in that state. This difference results in two problems:

- such atypical actions then lead to states unlike anything observed in the data,
- the policy that the system will use for future actions is different from that observed in the data, and

The first problem makes the $Q_{data}(s,a)$ estimates for the visited states highly unreliable, because we don't have data for these states. Because the future reward depends on the policy that the system will use in the future, the second problem means that the estimate $Q_{data}(s,a)$ is not even relevant to the expected future reward of the pure RL policy. We will return to these problems when we develop our proposed method in Section 2.4.

These problems are a result of the fact that we are training on a fixed data set, and therefore cannot generate new data that is appropriate for the new policy. The solution to these problems that is typically used in RL research is to generate new data as learning progresses and the policy changes, as discussed in Section 1.1. The RL system can thus explore the space of possible policies and states, generating new data that is relevant to each explored policy and its states. The problem with learning with policy exploration, even when using simulated users, is that it is not tractable with a large state space and action set. Consider that with 10^{386} states and 74 actions, there are $74^{10^{386}}$ possible policies. If we were able to explore policies at a rate of 1 policy a second, after 1 year we would have visited only one policy in every $74^{10^{385.6}}$ policies. Policy exploration algorithms are only partially random, so to some extent they can make accurate choices about which parts of the policy space to explore and which to ignore, but these numbers are indicative of the scale of the problem faced by policy exploration. In addition, experiments with a random policy achieved an average score of -66, showing that the vast majority of policies are very bad. This indicates that starting policy exploration with a random policy would require an extremely large amount of exploration to move from there to a policy which is as good as the policy found with the proposal discussed herein (which achieved a score of 140, out of a maximum 197). Therefore it is crucial that exploratory learning at least be initialized with a policy that we already know to be good. The method proposed in this article for learning a policy from a pre-existing corpus of dialogues can be used to find such an initial policy.

Given these problems with using RL with a fixed data set, an obvious alternative would be to simply train a policy to mimic the policies of the systems used to generate the data. One reason for training a policy, rather than using one of the original policies, is that learning allows us to merge the policies from all the different systems, which can lead to a better policy than any one system (as we will show in Section 3). Another reason is that learning results in a policy that generalizes from the original policies in interesting ways. Most notably, our learning method can be used to define a probabilistic policy, not just the (presumably) deterministic policies used to generate the data. A third reason could be (as in our case) that we do not have access to any of the original systems that generated the data. In some sense we can use learning to reverse engineer the systems.

We train a policy to mimic the policy observed in the data using supervised learning with linear function approximation. This "pure SL" policy simply selects the action a with the highest probability $P(a|s)$ of being chosen given the state s. We estimate $P(a|s)$ with linear function approximation, just as for $Q_{data}(s,a)$, except that a normalized

exponential function (a.k.a. "softmax") is used so that the result is a probability distribution over actions a.

$$P(a|s) \approx S_{data}(s,a) = \frac{\exp(f(s)^T w'_a)}{\sum_{a'} \exp(f(s)^T w'_{a'})} \quad (3)$$

This gives us a log-linear model, also known as a maximum entropy model. The parameters of this model (the w'_a) are trained using supervised learning on the COMMUNICATOR data. As with the Q-function, the use of linear function approximation means that we have estimates for $P(a|s)$ even for states s that have never occurred in the data, based on similar states that did occur.

2.4 A Hybrid Approach to RL

In this work we focus on solving the first of the two problems we have discussed, namely, preventing the system from straying into portions of the state space for which we do not have sufficient data. To do this, we propose a novel hybrid approach that combines RL with supervised learning. SL is used to model which actions will take the system into a portion of the state space for which we don't have sufficient data. RL is used to choose between the remaining actions. A discriminant function $Q_{hybrid}(s,a)$ is derived that combines these two criteria in a principled way. The resulting policy can be adjusted to be as similar as necessary to the policy in the data, thereby also addressing the second problem discussed previously.

As with the pure SL policy, supervised learning is used to model the policy that the systems in the data actually use. Because in general multiple policies were used, we model the data's policy as a probabilistic policy, using the estimate $S_{data}(s,a)$ of $P(a|s)$ presented in the previous section. $S_{data}(s,a)$ is an estimate of the probability that a random system selected from those that generated the data would choose action a given that it is in state s. Because we are using function approximation to learn $S_{data}(s,a)$ from the data, it will generalize (or "smooth") the policies actually used to generate the data so that similar states will allow similar sets of actions.[6]

The hybrid approach we have investigated is based on the assumption that the Q-function trained on the data is a poor model of the expected future reward for states in the portion of the state space not covered by the data. Thus we need an alternative method for estimating the future reward for these unobserved states. We have experimented with two such methods. The first method simply specifies a fixed reward U for these states. By setting this fixed reward to a low value, it amounts to a penalty for straying from the observed portion of the state space.

The second method estimated the reward for unobserved states by adding a fixed reward offset UO to the reward estimates for ending the dialogue immediately. This method compensates for the use of a dialogue-final reward scheme, where many things that the dialogue has already accomplished aren't reflected in the reward given so far. For example, in our reward scheme, filling a slot does not result in immediate reward, but instead results in reward at the end of the dialogue if it leads to a successful dialogue. The estimated reward for ending the dialogue immediately reflects how much

6 For this reason, we will get a probabilistic policy even if only a single deterministic policy is used to generate the data. This makes this method applicable even for data sets generated with a single deterministic prototype system.

reward is stored up in the state in this way. If the fixed reward added to this estimate is set to negative, then we can be sure that the reward estimated for unobserved states is always less than that for the best observed state, so this method also results in a penalty for straying from the observed portion of the state space.

Given an estimated reward u for unobserved states, the expected future reward is then the average between u for the cases where performing a in s leads to an unobserved state and the expected reward $Q_{data}(s,a)$ for the cases where it leads to an observed state. Formally, this average is a mixture of the estimate u with the estimate $Q_{data}(s,a)$, where the mixture coefficient is the probability $P_{observed}(s,a)$ that performing a in s will lead to an observed state.

$$\begin{aligned} &E_{d|s_i,a}[\textstyle\sum_{j>i} r(d,j)] \\ &\quad \approx Q_{data}(s,a)P_{observed}(s,a) + u(1 - P_{observed}(s,a)) \end{aligned} \tag{4}$$

Because this estimate of the expected future reward is only needed for choosing the next action given the current state s, we only need to estimate a function that discriminates between different actions in the same way as this estimate. To derive such a discriminant function, we first approximate $P_{observed}(s,a)$ with a first-order approximation in terms of the probability distribution in the data $P(s,a)$ and the size of the data set N, under the assumption that the number of possible state–action pairs is much larger than the size of the data set (so $P(s,a)N \ll 1$).

$$\begin{aligned} P_{observed}(s,a) &= 1 - (1 - P(s,a))^N \\ &\approx P(s,a)N \approx S_{data}(s,a)P(s)N \end{aligned} \tag{5}$$

Given this approximation, the discriminant function needs to order two actions a_1, a_2 in the same way as this estimate of the expected future reward.

$$\begin{aligned} &Q_{data}(s,a_1)S_{data}(s,a_1)P(s)N + u(1 - S_{data}(s,a_1)P(s)N) \\ &\quad \leq Q_{data}(s,a_2)S_{data}(s,a_2)P(s)N + u(1 - S_{data}(s,a_2)P(s)N) \\ &\textit{if and only if} \\ &S_{data}(s,a_1)(Q_{data}(s,a_1) - u) \leq S_{data}(s,a_2)(Q_{data}(s,a_2) - u) \end{aligned} \tag{6}$$

We call this discriminant function $Q_{hybrid}(s,a)$.

$$Q_{hybrid}(s,a) = S_{data}(s,a)(Q_{data}(s,a) - u) \tag{7}$$

We use this $Q_{hybrid}(s,a)$ function to choose the actions for our hybrid policy. By adjusting the value of the unobserved state penalty u, we can adjust the extent to which this model follows the supervised policy defined by $S_{data}(s,a)$ or the reinforcement learning policy defined by $Q_{data}(s,a)$. In particular, if u is very low, then maximizing $Q_{hybrid}(s,a)$ is equivalent to maximizing $S_{data}(s,a)$. Thus a very low u is equivalent to the policy that always chooses the most probable action, which we will call the "SL policy."

The procedure for training $Q_{hybrid}(s,a)$ is simply to train $Q_{data}(s,a)$ with RL and $S_{data}(s,a)$ with SL. These two models are then combined using Equation (7), given a value for u computed with one of the two methods presented previously. Both of these methods involve setting a constant that determines the relative importance of RL versus SL. In the next section we will empirically investigate good values for these constants.

3. Empirical Evaluation

We evaluate the trained dialogue management policies by running them against trained user simulations. The policies and the user simulations were trained using different parts of the annotated COMMUNICATOR data (using two-fold and five-fold cross validation). We compare our results against each other and against the performance of the eight COMMUNICATOR systems, using an evaluation metric discussed subsequently. The Information States for the simulated dialogues were computed with the same rules used to compute the Information States for the annotated data.

3.1 The Testing Setup

For these experiments, we restrict our attention to users who only want single-leg and return flight bookings. This allows us to do the evaluation using only the four essential slots included in both these types of bookings: origin city, destination city, departure date, and departure time. To achieve this restriction, we first selected all those COMMUNICATOR dialogues that consisted only of single-leg or return flight bookings. This subset contained 217 ATT dialogues, 116 BBN dialogues, 126 CMU dialogues, 159 Colorado dialogues, 77 IBM dialogues, 192 Lucent dialogues, 180 MIT dialogues, and 185 SRI dialogues, for a total of 1,252 dialogues (out of 1,683). This subset was used for evaluating the COMMUNICATOR systems and for training the user models. The system models were trained on the full set of dialogues, because they should not know the user's goals in advance. So, for each fold of the data, the user model was trained on only the single-leg and return dialogues from that fold and the system model was trained on the full set of dialogues from a subset of the remaining folds (one fold for the two-fold experiments and three folds for the five-fold experiment, as discussed subsequently).

The user models were trained in the same way as the $S_{data}(s, a)$ function for the pure SL model discussed in Section 2.3, using linear function approximation and a normalized exponential output function. The states that precede user actions are input as vectors of features virtually identical to those used for the system. However, unlike the action set for the system, the user only chooses one action per turn, and that action can include multiple ⟨Speech Act, Task⟩ pairs. The output of the model is a probability distribution over these actions. The user simulation selects an action randomly according to this distribution. We also trained a user model based on n-grams of user and system actions, which produced similar results in our testing (Georgila, Henderson, and Lemon 2006).

In our initial experiments with the hybrid policy, we found that it never closed the dialogue. We think that this was due to the system action (annotated in DATE) `meta_greeting_goodbye`, which is used both as the first action and as the last action of a dialogue. The hybrid policy expects this action to be chosen before it will close the dialogue, but the system never chooses this action at the end of a dialogue because it is so strongly associated with the beginning of the dialogue. This is an example of the limitations of linear function approximation, and our dependence on the previous COMMUNICATOR annotations. We could address this problem by splitting this action into two actions, one for "greeting" and one for "goodbye." But because we do not want to embark on the task of feature engineering at this stage, we have instead augmented the hybrid policy with a rule that closes the dialogue after the system chooses the action `offer`, to offer the user a flight. After this first flight offer, the user has one turn to reply, and then the dialogue is ended. For practical reasons we have also added rules

that close the dialogue after 100 states (i.e., total of user and system actions), and that release the turn if the system has done 10 actions in a row without releasing the turn.

3.2 The Evaluation Metrics

To evaluate the success of a dialogue, we take the final state of the dialogue and use it to compute a scoring function. We want the scoring function to be similar to the reward we compute from the quality measures provided with the COMMUNICATOR data (e.g., the user questionnaires), but because we do not have these quality measures for the simulated dialogues, we cannot use the exact same reward function. When we compare the hybrid policy against the COMMUNICATOR systems, we apply the same scoring function to both types of dialogues so that we have a comparable evaluation metric for both.

Because currently we are only considering users who only want single-leg or return flight bookings, the scoring function only looks at the four essential slots for these bookings: origin city, destination city, departure date, and departure time. We give 25 points for each slot that is filled, plus another 25 points for each slot that is also confirmed. We also deduct 1 point for each action performed by the system, to penalize longer dialogues. Thus the maximum possible score is 197 (i.e., 200 minus 3 system actions: ask for all the user information in one action, then confirm all the four slots in one action and offer a flight).

The motivation behind this evaluation metric is that confirmed slots are more likely to be correct than slots that are just filled. If we view the score as proportional to the probability that a slot is filled correctly, then this scoring assumes that confirmed slots are twice as likely to be correct. Although other scoring metrics are clearly possible, this one is a simple and reasonable approximation of the relative expected correctness of confirmed versus non-confirmed information in dialogue systems. On the other hand, none of our conclusions depend on this exact scoring function, as indicated by results for the "no-conf" version of our scoring function (discussed subsequently), which ignores confirmations.

When combining the scores for different slots, we do not try to model the all-or-nothing nature of the COMMUNICATOR task-completion quality measures, but instead sum the scores for the individual slots. This sum makes our scoring metric value partial completions more highly, but inspection of the distributions of scores indicates that this difference does not favor either the hybrid policy or the original COMMUNICATOR systems.

Although this evaluation metric could reflect the relative quality of individual dialogues more accurately, we believe it provides a good measure of the relative quality of the systems we wish to compare. First, the exact same metric is applied to every system. Additional information that we have for some systems, but not all, is not used (e.g., the COMMUNICATOR user questionnaires, which we do not have for simulated dialogues). Second, the systems are being run against approximately equivalent users. The user simulation is trained on exactly the same user actions that are used to evaluate the COMMUNICATOR systems, so the user simulations mimic exactly these users. In particular, the simulation is able to mimic the effects of speech recognition errors, because it is just as likely as the real users to disagree with a confirmation or provide a new value for a previously filled slot. The nature of the simulation model may make it systematically different from real users in some way, but we know of no argument for why this would bias our results in favor of one system or another.

One concern about this evaluation metric is that it does not reflect the quality of the speech recognizer being used by the system. If a system has a good speech recognizer, then it may not be necessary for it to confirm a slot value, but our scoring function will still penalize it for not confirming. This would certainly be a problem if this metric were to be used to compare different systems within the COMMUNICATOR data set. However, the intention of the metric is simply to facilitate comparisons between different versions of our proposed system, and between our proposed systems and those in the data. Because the user simulations are trained on the COMMUNICATOR data, they simulate speech recognition errors at the same rate as the data, thereby controlling for the quality of the speech recognizer.

Nonetheless, it is worth considering another evaluation metric that does not penalize for missing confirmations. For this reason we also evaluate the different systems based on their scores for only filled slots and length, which we call the "no-conf" score.

3.3 The Influence of Reinforcement Learning

In our first set of experiments, we evaluated the success of our hybrid policy relative to the performance of the pure reinforcement learning policy and the pure supervised learning policy. We also investigated how to best set the parameters for combining the supervised and reinforcement learning policies in a hybrid policy.

We first compared the two proposed hybrid methods using two-fold cross validation. We trained models of both $Q_{data}(s,a)$ and $S_{data}(s,a)$, and then used them to define policies. We trained both models for 100 iterations through the training portion of the data, at which point there was little change in the training error. We trained $Q_{data}(s,a)$ using SARSA(λ) with $\lambda = 0.9$. This training was repeated twice, once for each fold of the complete data set. The reinforcement learning policy uses only $Q_{data}(s,a)$, the SL policy uses only $S_{data}(s,a)$, and the hybrid policies combine the two using Equation (7). For the hybrid policies, we used the two methods for estimating the unobserved state penalty u and various values for the fixed reward U or reward offset UO.

During testing, each policy was run for 2,000 dialogues against a linear function approximation user model trained on the opposite half of the data. The final state for each one of these dialogues was then fed through the scoring function and averaged across dialogues and across data halves. The results are plotted in Figure 2. To allow direct comparisons between the different values of U and UO, these scores are plotted against the proportion of decisions that are different from that which the pure SL policy would choose. Thus the SL policy (average reward 139.8) is plotted at 0 (which is

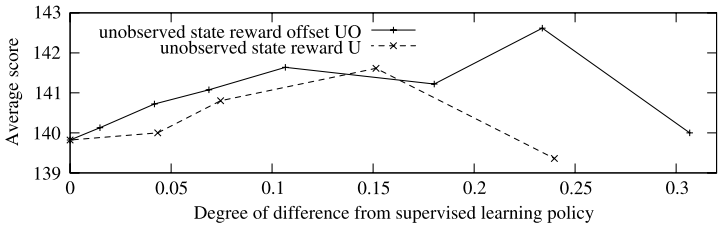

Figure 2
Average dialogue score plotted against the proportion of decisions that diverge from the SL policy, for different values of the unobservable state reward U and reward offset UO. Averages over two folds, 2,000 dialogues per fold.

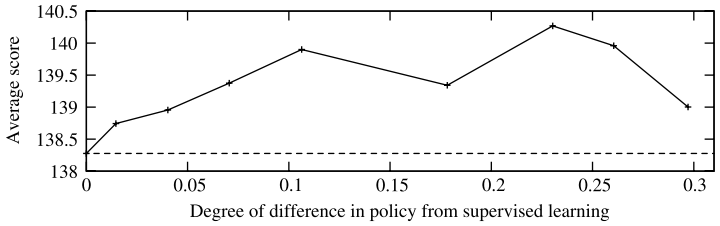

Figure 3
Average dialogue score plotted against the proportion of decisions that diverge from the SL policy, for different values of the unobservable state reward offset *UO*. Averages over five folds, 2,000 dialogues per fold.

equivalent to a large negative *U* or *UO*). Additional points for the hybrid policies are shown for (from left to right, respectively) $U = 0$, 40, 80, and 100, and $UO =$ –300, –100, –60, –40, –20, –10, and 0. The pure reinforcement learning policy is not shown because its average score falls well below the bottom of the graph, at 44.4.

Figure 2 indicates that, for both hybrid methods, adding some influence from RL increases performance over pure SL, but too much RL results in degradation. Using a reward offset *UO* for u generally does better than a fixed reward *U*, and allows a greater influence from RL before degradation.

We found that the results for our two folds were very different,[7] so we repeated the experiments using five-fold cross validation, where the dialogues from each system were split randomly (rather than chronologically).[8] For each fold, we trained models of both $Q_{data}(s,a)$ and $S_{data}(s,a)$ on three of the folds, using a fourth fold to decide when to stop training. The fifth fold was then used to train a linear function approximation user model, which was used to generate 2,000 simulated dialogues. Combining the five folds, this gave us 10,000 dialogues per model. Because, in the previous experiments, using a reward offset *UO* performed better than using a fixed reward *U*, we only tested models using different values of the reward offset *UO*.

The validation performance of the trained models for $Q_{data}(s,a)$ and $S_{data}(s,a)$ performed similarly across the different splits. Taken together, the models of $S_{data}(s,a)$ had a perplexity of 4.4. Intuitively, this means that the supervised models were able to narrow down the list of possible actions from 74 to about 4 choices, on average. This suggests that the ISU representation of state is doing a good job of representing the information being used by the systems to make dialogue management decisions, but that there is still a good amount of uncaptured variability. Presumably most of this variability is due to differences between the policies for the different systems. The models of $Q_{data}(s,a)$ had a mean squared error of 8,242, whose square root is 91. This measure is harder to interpret because it is dominated by large errors, but suggests that the expected future reward is rather hard to predict, as is to be expected.

Figure 3 shows the average scores for the pure SL policy (at 0) and for hybrid policies (from left to right) with $UO = -300, -100, -60, -40, -20, -10, -5$ and 0. The

7 For the two-fold experiments, the data were split by putting the first half of the dialogues for each system in one fold and the second half in the other, under the constraint that no user had dialogues in more than one fold. It appears that the users that were run at the beginning of the 2001 COMMUNICATOR data collection were very different from those run at the end.

8 To be more precise, for each system we split the set of users randomly into five groups. Then all the dialogues for a given group of users were put in the same fold.

hybrid policies perform consistently better than the SL policy. The difference between the hybrid policy and the SL policy is statistically significant at the 5% level for the three best hybrid policies tested ($p < 0.01$ for $UO = -40$, $p < 0.001$ for $UO = -10$, and $p < 0.007$ for $UO = -5$). If we combine all the tested hybrid policies together, then their average score (139.4) is also significantly better than the SL policy ($p < 0.014$). All these results are significantly better than the average score of the pure RL policy (34.9).

3.4 Comparisons with COMMUNICATOR Systems

In our second set of experiments, we evaluated the success of our learned policies relative to the performance of the COMMUNICATOR systems that they were trained on. To evaluate the performance of the COMMUNICATOR systems, we extracted final states from all the dialogues that only contain single-leg or return flight bookings and fed them through the scoring function. The average scores are shown in Tables 3 and 4, along with the average scores for the pure SL policy, the pure RL policy, and the best hybrid policy ($UO = -10$). The total score, the score excluding confirmations, and the three components of the total score are shown.

Table 3 shows the results computed from the complete dialogues. These results show a clear advantage for the hybrid policy over the average across the COMMUNICATOR systems, as well as over each individual COMMUNICATOR system. In particular, the hybrid policy uses fewer steps. Because the number of steps is doubtless affected by the hybrid policy's built-in strategy of stopping the dialogue after the first flight offer, we also evaluated the performance of the COMMUNICATOR systems if we also stopped these dialogues after the first flight offer, shown in Table 4. The COMMUNICATOR systems do better when stopped at the first flight offer, but still their average ("all COMMUNICATOR") is not nearly as good as the hybrid or SL policies, under all measures.

Although the average score of the COMMUNICATOR systems in Table 4 is well below those of the hybrid and SL policies, under this measure the single best system (BBN) beats our proposed system. Also, if we ignore confirmations (the "no-conf" measure),

Table 3
The average scores from the different systems for single-leg and return dialogues, the score excluding confirmations, and the three components of these scores.

System	Total score	No-conf score	Filled slots	Confirmed slots	Length penalty
hybrid RL/SL	140.3	70.3	88.0	70.0	−17.7
pure SL	138.3	69.2	89.2	69.1	−20.0
pure RL	34.9	25.6	56.9	8.3	−31.3
all COMMUNICATOR	103.6	40.6	85.0	63.0	−44.4
SRI	115.3	50.5	83.4	64.9	−32.9
MIT	114.3	43.2	87.1	71.1	−43.9
LUC	110.3	36.1	91.1	74.1	−55.0
COL	105.9	47.0	90.6	59.0	−43.6
BBN	102.4	27.1	82.5	75.2	−55.4
ATT	94.0	38.7	78.3	55.3	−39.6
CMU	92.1	24.0	81.7	68.1	−57.7
IBM	77.0	61.8	85.4	15.3	−23.6

Table 4
The average scores after the first flight offer for single-leg and return dialogues, the score excluding confirmations, and the three components of these scores.

System	Total score	No-conf score	Filled slots	Confirmed slots	Length penalty
hybrid RL/SL	140.3	70.3	88.0	70.0	−17.7
pure SL	138.3	69.2	89.2	69.1	−20.0
pure RL	34.9	25.6	56.9	8.3	−31.3
all COMMUNICATOR	127.1	63.2	84.5	63.9	−21.3
BBN	148.9	73.2	88.6	75.6	−15.4
LUC	138.5	59.1	91.1	79.4	−32.1
MIT	136.4	66.9	82.8	69.4	−15.9
COL	132.9	71.4	89.9	61.5	−18.6
SRI	128.2	61.7	84.2	66.5	−22.5
CMU	123.8	58.7	77.2	65.1	−18.5
ATT	109.1	53.6	78.3	55.4	−24.7
IBM	86.4	71.2	85.1	15.3	−13.9

then three of the individual systems beat our proposed system by small amounts. However, as discussed in Section 3.2, our evaluation methodology is not really appropriate for comparing against individual COMMUNICATOR systems, due to likely differences in speech recognition performance across systems. To test this explanation, we looked at the word error rates for the speech recognition outputs for the different systems. BBN has the highest percentage of user utterances with no speech recognition errors (79%, versus an average of 66%), and the second lowest average word error rate (12.1 versus an average of 22.1). Because our simulated users simulate speech recognition errors at the average rate, the difference in performance between BBN and our systems could easily be explained simply by differences in the speech recognizers, and not differences in the dialogue management policies.

3.5 Discussion

The most obvious conclusion to draw from these results is not a surprising one: Pure reinforcement learning with such a huge state space and such limited data does not perform well. Given the pure RL policy's score of 34.9, all the policies in Figure 3 and all the COMMUNICATOR systems in Tables 3 and 4 perform better by quite a large margin. Inspection of the dialogues indicates that the pure RL policy does not result in a coherent sequence of actions. This policy tends to choose actions that are associated with the end of the dialogue, even at the beginning of the dialogue. Perhaps this is because these actions are only chosen by the COMMUNICATOR systems during relatively successful dialogues. This policy also tends to repeat the same actions many times, for example repeatedly requesting information even after the user has supplied this information. These phenomena are examples of the problem we used to motivate our hybrid learning method, in that they both involve state–action pairs that the learner would never have seen in the COMMUNICATOR training data.

Given the disappointing performance of the pure RL policy, it is surprising that our hybrid policies outperform the pure SL policy, as shown in Figures 2 and 3. Though the increase in performance is small, it is statistically significant, and consistent across the

two hybrid methods and across a range of degrees of influence from RL.[9] This indicates that our hybrid policies are succeeding in getting useful information from the results of reinforcement learning, even under these extremely difficult circumstances. Perhaps, under less severe circumstances for RL, a greater gain can be achieved with hybrid policies. For the second hybrid policy (unobserved state reward offset), the fact that the best result was achieved with a *UO* value ($UO = -10$) that is very close to the theoretical limit of this method ($UO = 0$) suggests that future improvements to this method could result in even more useful information being extracted from the RL policy.

The different components of the scoring function give some indication of how the hybrid policies differ from the SL policy. As indicated in the top two rows of Table 4, the hybrid policies mostly improve over the SL policy in dialogue length, with a slight increase in confirmed slots and a slight decrease in filled slots.

One striking conclusion from the results comparing the learned policies to the policies of the COMMUNICATOR systems, shown in Tables 3 and 4, is that the learned policies score better than the policies they were trained on. This is particularly surprising for the pure SL policy, given that this policy is simply trying to mimic the behavior of these same systems. This can be explained by the fact that the SL policy is the result of merging all policies of the COMMUNICATOR systems. Thus it can be thought of as a form of multi-version system, where decisions are made based on what the majority of systems would do.[10] Multi-version systems are well known to perform better than their component systems, because the mistakes tend to be different across the different component systems. They remove errors made by any one system that are not shared by most of the other systems.

The good performance of the SL policy compared to the COMMUNICATOR systems makes the better performance of the hybrid policies even more impressive. As shown on the x axis of Figure 3, the best hybrid systems choose a different action from the SL policy about one action out of four. Despite the good performance of the action chosen by the SL policy, RL is able to (on average) find a better action by looking at the rewards achieved by the systems in the data when they chose those actions in similar states. By following different systems' choices at different points in the dialogue, the learned policy can potentially perform better than any individual system. Although our current evaluation methodology is not fine-grained enough to determine if this is being achieved, the most promising aspect of applying RL to fixed data sets is in learning to combine the best aspects of each system in the data set.

Although we believe that these results provide an accurate picture of the relative strengths of the different types of systems we compare, it should be noted that the reliance on evaluation with simulated dialogues inevitably leads to some lack of precision in the evaluation. All these results are computed with users who have the same goal (booking a return flight) and with an evaluation metric that only looks at dialogue length and whether the four main slots were filled and (optionally) confirmed. On the

9 We previously reported results that showed that adding influence from reinforcement learning always degraded performance slightly compared to the pure SL policy (Henderson, Lemon, and Georgila 2005). However, these results were obtained with a preliminary version of the data annotation, which gave a less accurate indication of when slots were filled and confirmed. The scores we are achieving with the new data annotation (Georgila et al. submitted) are all higher than those reported in Henderson, Lemon, and Geogila (2005), including the scores calculated from the data for the COMMUNICATOR systems themselves.

10 To be more technically accurate, we can think of the SL policy as in effect asking each COMMUNICATOR system for a probability distribution over state–action pairs for the current state, summing these probabilities across systems, and choosing the action with the highest probability.

other hand, all the systems were trained to handle a more complicated task than this, including multi-leg flights, hotel bookings, and rental-car bookings. They were also designed or trained to complete the task, rather than to fill the slots. Therefore the evaluation does not reflect all the capabilities or behaviors of the systems. However, there is no apparent reason to believe that this fact biases our results towards one type of system or another. This claim is easiest to support for the comparisons between the hybrid method and the two trained baselines, pure RL and pure SL. For all these systems, the same data was used to train the systems, the same user models were used to generate simulated dialogues, and the same evaluation metric was applied to these simulated dialogues. For the comparisons between the hybrid method and the COMMUNICATOR systems, only the evaluation metric is exactly the same, but the user models used for testing the hybrid method were trained to mimic exactly the users in the dialogues used to evaluate the COMMUNICATOR systems. Because we know of no evaluation bias introduced when moving from real users to their simulation, we conclude that this comparison is also indicative of the relative performance of these two types of systems (particularly given the size of the improvement).

A more general methodological objection could be raised against any evaluation that uses simulated users. Despite the substantial amount of dialogue system work that has relied on simulated users (e.g., Scheffler and Young 2002; Pietquin 2004; Georgila, Henderson, and Lemon 2006; Schatzmann et al. 2006), to date there has not been a systematic experiment that validates this methodology against results from human users. However, in related work (Lemon, Georgila, and Henderson 2006), we have demonstrated that a hybrid policy learned as proposed in this article performs better than a state-of-the-art hand-coded system in experiments with human users. The experiments were done using the "Town Information" multimodal dialogue system of Lemon et at. (2006) and Lemon, Georgila, and Stuttle (2005). The hybrid policy reported here (trained on the COMMUNICATOR data) was ported to this domain, and then evaluated with human subjects. The learned policy achieved an average gain in perceived task completion of 14.2% (from 67.6% to 81.8% at $p < 0.03$) compared to a state-of-the-art hand-coded system (Lemon, Georgila, and Henderson 2006). This demonstrates that a policy that performs well in simulation also performs well in real dialogues.[11]

These experiments demonstrate improvements given an initial fixed data set which has been generated from existing systems. For applications where there are no existing systems, an alternative would be to generate the initial data with a Wizard-of-Oz experiment, where a human plays the part of the system, as explored by Williams and Young (2003) and Rieser and Lemon (2006b). The methods proposed in this article can be used to train a policy from such data without having to first build an initial system.

4. Conclusions

In this article, we have investigated how reinforcement learning can be applied to learn dialogue management policies with large action sets and very large state spaces given only a fixed data set of dialogues. Under a variety of metrics, our proposed hybrid reinforcement learning method outperforms both a policy trained with standard RL and a

11 Future work is to port the hand-coded policy back to the COMMUNICATOR domain for use in simulation. This will investigate whether a relative improvement in simulated dialogues translates into a relative improvement in real dialogues.

policy trained with supervised learning, as well as the COMMUNICATOR systems which generated the data it was trained on. This performance is achieved despite the extremely challenging task, with 74 actions to choose between, over 10^{386} possible states, and very few hand-coded policy decisions. The two main features of our model that make this possible are the incorporation of supervised learning into a reinforcement learning model, and the use of linear function approximation with state features provided by the Information State Update approach to dialogue management. The supervised learning is used to avoid states not covered by the data set, and the linear function approximation is used to handle the very large state spaces.

With such a large space of possible state–action pairs, and therefore a huge policy space, pure reinforcement learning would require an enormous amount of data to find good policies. We have succeeded in using RL with fairly small data sets of only around 1,000 dialogues (in the portion used for training). This is achieved by using supervised learning to model when an action would lead to a state for which we do not have enough data. We proposed two methods for estimating a default value for these unseen states, and derived a principled way to combine this value with the value estimated by RL, using the probability provided by SL to weight this combination. This gave us two hybrid RL/SL methods, both of which outperform both the RL and SL policies alone. The best hybrid policy performs 302% better than the standard RL policy, and 1.4% better than the SL policy, according to our automatic evaluation method. In addition, according to our automatic evaluation method, the hybrid RL/SL policy outperforms the systems used to generate the data. The best hybrid policy improves over the average COMMUNICATOR system policy by 10% on our metric. This good performance has been corroborated in separate experiments with human subjects (Lemon, Georgila, and Henderson 2006), where the learned policy outperforms a state-of-the-art hand-coded system.

The success of the hybrid method (and of pure supervised learning) on this challenging task indicates that linear function approximation is a viable approach to the very large state spaces produced by the ISU framework. It also demonstrates the utility of a feature-based representation of states, such as that used in the ISU approach. Further improvement should be possible by tailoring the representation of states and actions based on our experience so far (e.g., by including information about specific sequences of moves), and by using automatic feature selection techniques. We should also be able to get some improvement from more sophisticated function approximation methods, such as kernel-based methods.

The next step is to better exploit the advantages of reinforcement learning. One promising approach is to apply RL while running the learned policy against simulated users, thereby allowing RL to explore parts of the policy and state spaces that are not included in the COMMUNICATOR data. The hybrid policy we have learned on the COMMUNICATOR data is a good starting point for this exploration. Also, the supervised component within the hybrid system can be used to constrain the range of policies that need to be explored when training the RL component. All of these advances will improve techniques for bootstrapping and automatic optimization of dialogue management policies from limited initial data sets.

Acknowledgments

This work was partially supported by the European Commission under the FP6 project "TALK: Talk and Look, Tools for Ambient Linguistic Knowledge" (507802) and the FP7 project "CLASSIC: Computational Learning in Adaptive Systems for Spoken Conversation" (216594), by the EPSRC under grant EP/E019501/1, and by SHEFC HR04016—Wellcome Trust VIP Award.

We thank Johanna Moore for proposing the use of the COMMUNICATOR data set for this work.

References

Bohlin, Peter, Robin Cooper, Elisabet Engdahl, and Staffan Larsson. 1999. Information states and dialog move engines. *Electronic Transactions in AI*, 3(9). Available at `www.ep.liu.se/ej/etai/1999/D/`.

Bos, Johan, Ewan Klein, Oliver Lemon, and Tetsushi Oka. 2003. DIPPER: Description and formalisation of an information-state update dialogue system architecture. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialogue*, pages 115–124, Sapporo.

Cheyer, Adam and David Martin. 2001. The open agent architecture. *Journal of Autonomous Agents and Multi-Agent Systems*, 4(1/2):143–148.

Denecke, Matthias, Kohji Dohsaka, and Mikio Nakano, 2005. Fast reinforcement learning of dialogue policies using stable function approximation. In K. Y. Su, J. Tsujii, J.-H. Lee, and O. Y. Kwong, *Natural Language Processing, IJCNLP 2004.* Springer, Berlin, pages 1–11.

Frampton, Matthew and Oliver Lemon. 2005. Reinforcement learning of dialogue strategies using the user's last dialogue act. In *Proceedings of the 4th Workshop on Knowledge and Reasoning in Practical Dialog Systems, International Joint Conference on Artificial Intelligence (IJCAI)*, pages 83–90, Edinburgh.

Frampton, Matthew and Oliver Lemon. 2006. Learning more effective dialogue strategies using limited dialogue move features. In *Proceedings of the 44th Meeting of the Association for Computational Linguistics*, pages 185–192, Sydney.

Gabsdil, Malte and Oliver Lemon. 2004. Combining acoustic and pragmatic features to predict recognition performance in spoken dialogue systems. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, pages 344–351, Barcelona.

Georgila, Kallirroi, James Henderson, and Oliver Lemon. 2005. Learning user simulations for Information State Update dialogue systems. In *Proceedings of the 9th European Conference on Speech Communication and Technology (Interspeech – Eurospeech)*, pages 893–896, Lisbon.

Georgila, Kallirroi, James Henderson, and Oliver Lemon. 2006. User simulation for spoken dialogue systems: Learning and evaluation. In *Proceedings of the 9th International Conference on Spoken Language Processing (Interspeech – ICSLP)*, pages 1065–1068, Pittsburgh, PA.

Georgila, Kallirroi, Oliver Lemon, and James Henderson. 2005. Automatic annotation of COMMUNICATOR dialogue data for learning dialogue strategies and user simulations. In *Proceedings of the Ninth Workshop on the Semantics and Pragmatics of Dialogue (SEMDIAL)*, pages 61–68, Nancy.

Georgila, Kallirroi, Oliver Lemon, James Henderson, and Johanna Moore. (submitted). Automatic annotation of context and speech acts for dialogue corpora.

Goddeau, D. and J. Pineau. 2000. Fast reinforcement learning of dialog strategies. In *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages II–1233–1236, Istanbul.

Henderson, James, Oliver Lemon, and Kallirroi Georgila. 2005. Hybrid reinforcement/supervised learning for dialogue policies from COMMUNICATOR data. In *Proceedings of the 4th Workshop on Knowledge and Reasoning in Practical Dialog Systems, International Joint Conference on Artificial Intelligence (IJCAI)*, pages 68–75, Edinburgh.

Larsson, Staffan and David Traum. 2000. Information state and dialogue management in the TRINDI Dialogue Move Engine Toolkit. *Natural Language Engineering*, 6(3–4):323–340.

Lemon, Oliver, Kallirroi Georgila, and James Henderson. 2006. Evaluating effectiveness and portability of reinforcement learned dialogue strategies with real users: the TALK TownInfo evaluation. In *Proceedings of the IEEE/ACL 2006 Workshop on Spoken Language Technology*, pages 178–181, Aruba.

Lemon, Oliver, Kallirroi Georgila, James Henderson, Malte Gabsdil, Ivan Meza-Ruiz, and Steve Young. 2005. Integration of learning and adaptivity with the ISU approach. Technical Report D4.1, TALK Project.

Lemon, Oliver, Kallirroi Georgila, James Henderson, and Matthew Stuttle. 2006. An ISU dialogue system exhibiting reinforcement learning of dialogue policies: generic slot-filling in the TALK

in-car system. In *Proceedings of the Demonstrations of EACL*, pages 119–122, Trento.
Lemon, Oliver, Kallirroi Georgila, and Matthew Stuttle. 2005. Showcase exhibiting reinforcement learning for dialogue strategies in the in-car domain. Technical Report D4.2, TALK Project.
Levin, Esther and Roberto Pieraccini. 1997. A stochastic model of computer-human interaction for learning dialogue strategies. In *Proceedings of the 5th European Conference on Speech Communication and Technology (Interspeech – Eurospeech)*, pages 1883–1886, Rhodes.
Levin, Esther, Roberto Pieraccini, and Wieland Eckert. 2000. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, 8(1):11–23.
Pietquin, Olivier. 2004. *A Framework for Unsupervised Learning of Dialogue Strategies*. Presses Universitaires de Louvain, SIMILAR Collection.
Pietquin, Olivier and Thierry Dutoit. 2006a. Dynamic Bayesian networks for NLU simulation with application to dialog optimal strategy learning. In *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 49–52, Toulouse.
Pietquin, Olivier and Thierry Dutoit. 2006b. A probabilistic framework for dialog simulation and optimal strategy learning. *IEEE Transactions on Speech and Audio Processing*, 14(2):589–599.
Rieser, Verena and Oliver Lemon. 2006a. Cluster-based user simulations for learning dialogue strategies and the SUPER evaluation metric. In *Proceedings of the 9th International Conference on Spoken Language Processing (Interspeech – ICSLP)*, pages 1766–1769, Pittsburgh, PA.
Rieser, Verena and Oliver Lemon. 2006b. Using logistic regression to initialise reinforcement-learning-based dialogue systems. In *Proceedings of the IEEE/ACL 2006 Workshop on Spoken Language Technology*, pages 190–193, Aruba.
Rieser, Verena and Oliver Lemon. 2006c. Using machine learning to explore human multimodal clarification strategies. In *Proceedings of the Poster Session of the 44th Meeting of the Association for Computational Linguistics*, pages 659–666, Sydney.
Roy, Nicholas, Joelle Pineau, and Sebastian Thrun. 2000. Spoken dialog management for robots. In *Proceedings of the 38th Meeting of the Association for Computational Linguistics*, pages 93–100, Hong Kong.
Schatzmann, Jost, Kallirroi Georgila, and Steve Young. 2005. Quantitative evaluation of user simulation techniques for spoken dialogue systems. In *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*, pages 45–54, Lisbon.
Schatzmann, Jost, Matthew N. Stuttle, Karl Weilhammer, and Steve Young. 2005. Effects of the user model on simulation-based learning of dialogue strategies. In *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*, pages 220–225, San Juan, Puerto Rico.
Schatzmann, Jost, Karl Weilhammer, Matthew N. Stuttle, and Steve Young. 2006. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The Knowledge Engineering Review*, 21:97–126.
Scheffler, Konrad and Steve Young. 2001. Corpus-based dialogue simulation for automatic strategy learning and evaluation. In *Proceedings of the NAACL Workshop on Adaptation in Dialogue Systems*, pages 64–70, Pittsburgh, PA.
Scheffler, Konrad and Steve Young. 2002. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proceedings of the Human Language Technology Conference*, pages 12–19, San Diego, CA.
Shawe-Taylor, John and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.
Singh, Satinder, Michael Kearns, Diane Litman, and Marilyn Walker. 2000a. Empirical evaluation of a reinforcement learning dialogue system. In *Proceedings of the AAAI*, pages 645–651, Whistler.
Singh, Satinder, Michael Kearns, Diane Litman, and Marilyn Walker. 2000b. Reinforcement learning for spoken dialogue systems. In *Advances in Neural Information Processing Systems*, 12:956–962.
Singh, Satinder, Diane Litman, Michael Kearns, and Marilyn Walker. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research (JAIR)*, 16:105–133.
Sutton, Richard and Andrew Barto. 1998. *Reinforcement Learning*. MIT Press, Cambridge, MA.

Walker, M., J. Aberdeen, J. Boland, E. Bratt, J. Garofolo, L. Hirschman, A. Le, S. Lee, S. Narayanan, K. Papineni, B. Pellom, B. Polifroni, A. Potamianos, P. Prabhu, A. Rudnicky, G. Sanders, S. Seneff, D. Stallard, and S. Whittaker. 2001. DARPA communicator dialog travel planning systems: The June 2000 data collection. In *Proceedings of the 7th European Conference on Speech Communication and Technology (Interspeech – Eurospeech)*, pages 1371–1374, Aalborg.

Walker, M. and R. Passonneau. 2001. DATE: A dialogue act tagging scheme for evaluation of spoken dialogue systems. In *Proceedings of the Human Language Technology Conference*, pages 1–8, San Diego, CA.

Walker, M., A. Rudnicky, J. Aberdeen, E. Bratt, J. Garofolo, H. Hastie, A. Le, B. Pellom, A. Potamianos, R. Passonneau, R. Prasad, S. Roukos, G. Sanders, S. Seneff, D. Stallard, and S. Whittaker. 2002a. DARPA Communicator Evaluation: Progress from 2000 to 2001. In *Proceedings of the 7th International Conference on Spoken Language Processing (Interspeech – ICSLP)*, pages 273–276, Denver, CO.

Walker, M., A. Rudnicky, R. Prasad, J. Aberdeen, E. Bratt, J. Garofolo, H. Hastie, A. Le, B. Pellom, A. Potamianos, R. Passonneau, S. Roukos, G. Sanders, S. Seneff, and D. Stallard. 2002b. DARPA Communicator: Cross-system results for the 2001 evaluation. In *Proceedings of the 7th International Conference on Spoken Language Processing (Interspeech – ICSLP)*, pages 269–272, Denver, CO.

Walker, Marilyn A., Jeanne C. Fromer, and Shrikanth Narayanan. 1998. Learning optimal dialogue strategies: A case study of a spoken dialogue agent for email. In *Proceedings of the 17th International Conference on Computational Linguistics*, pages 1345–1351, Montreal.

Walker, Marilyn A., Candace A. Kamm, and Diane J. Litman. 2000. Towards developing general models of usability with PARADISE. *Natural Language Engineering*, 6(3):363–377.

Walker, Marilyn A., Rebecca J. Passonneau, and Julie E. Boland. 2001. Quantitative and qualitative evaluation of DARPA Communicator spoken dialogue systems. In *Proceedings of the 39th Meeting of the Association for Computational Linguistics*, pages 515–522, Toulouse.

Williams, Jason. 2007. A method for evaluating and comparing user simulations: The Cramer-von Mises divergence. In *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*, pages 508–513, Kyoto.

Williams, Jason, Pascal Poupart, and Steve Young. 2005a. Factored partially observable Markov decision processes for dialogue management. In *Proceedings of the 4th Workshop on Knowledge and Reasoning in Practical Dialog Systems, International Joint Conference on Artificial Intelligence (IJCAI)*, pages 76–82, Edinburgh.

Williams, Jason, Pascal Poupart, and Steve Young. 2005b. Partially observable Markov decision processes with continuous observations for dialogue management. In *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*, pages 25–34, Lisbon.

Williams, Jason and Steve Young. 2003. Using Wizard-of-Oz simulations to bootstrap reinforcement-learning-based dialog management systems. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialogue*, pages 135–139, Sapporo.

Williams, Jason and Steve Young. 2005. Scaling up POMDPs for dialog management: The "Summary POMDP" method. In *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*, pages 177–182, San Juan, Puerto Rico.

Young, Steve. 2000. Probabilistic methods in spoken dialogue systems. *Philosophical Transactions of the Royal Society (Series A)*, 358(1769):1389–1402.