Hybrid Reinforcement/Supervised Learning for Dialogue Policies from COMMUNICATOR data

James Henderson and **Oliver Lemon** and **Kallirroi Georgila**
School of Informatics
University of Edinburgh
2 Buccleuch Place, Edinburgh EH8 9LW, United Kingdom
james.henderson@ed.ac.uk

Abstract

We propose a method for learning dialogue management policies from a fixed dataset. The method is designed for use with "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 very large policy space. To address the problem that any fixed dataset will only provide information about small portions of these state and policy spaces, we propose a hybrid model which combines reinforcement learning (RL) with supervised learning. The reinforcement learning is used to optimise a measure of dialogue reward, while the supervised learning is used to restrict the learnt policy to the portion of the space for which we have data. Linear function approximation is used to handle the large state space efficiently. We trained this model on a subset of the COMMUNICATOR corpus, to which we have added annotations for user actions and Information States. When tested with a user simulation trained on the same data, our model outperforms all the systems in the COMMUNICATOR data (it scores 37% higher than the best COMMUNICATOR system). All of these advances will improve techniques for bootstrapping and automatic optimisation of dialogue management policies from limited initial datasets.

1 Introduction

We investigate using a fixed corpus of dialogues to automatically optimise dialogue systems which have rich representations of dialogue context. The "Information State Update" (ISU) approach to dialogue [Larsson and Traum, 2000] employs such representations of dialogue context for flexible dialogue management, and the question arises as to whether dialogue management policies can be learnt [Levin and Pieraccini, 1997] for such systems. We focus on learning with a fixed corpus of dialogues because dialogue corpora are very expensive to produce, and it is often not practical to produce new dialogues during the course of learning. Even if dialogues can be automatically generated with simulated users, training on simulated dialogues does not replace the need to fully exploit the real data.

Previous work on learning dialogue management policies has focused on small state spaces and small sets of actions to choose between [Singh *et al.*, 2002; Levin *et al.*, 2000; Scheffler and Young, 2002]. They use reinforcement learning (RL) to find a policy which optimises a reward function over dialogues. In this paper we address the ambitious task of learning to choose between a relatively large number of actions (70 in our experiments), with a very large state space (over 10^{87} states are theoretically possible), given a fairly small corpus of dialogues (697 in our experiments). We use linear function approximation to handle the large state space, but this does not address the difficulty of searching for an optimal policy in the huge space of possible policies.

Given the huge policy space, any fixed dataset will only provide information about a small portion of this space. To address this problem, we propose a hybrid learning model which combines reinforcement learning (RL) with supervised learning. RL is used to optimise a measure of dialogue reward, while supervised learning is used to restrict the learnt policy to the portion of the space for which we have data. When trained on a subset of the COMMUNICATOR corpus [Walker *et al.*, 2001a; 2002] and tested with a user simulation trained on the same data, this model outperforms all the systems in the COMMUNICATOR data. When the relative importance of the RL component and the supervised learning component are adjusted, we currently find that a purely supervised model performs the best. However, for a range of degrees of influence of RL, the hybrid system still performs better than the COMMUNICATOR systems. In this paper, we first discuss the annotations we have added to the COMMUNICATOR data, then present the proposed learning method, and then present our evaluation method and the results.

2 Automatic Annotation of the COMMUNICATOR Data

The COMMUNICATOR corpora (2000 [Walker *et al.*, 2001a] and 2001 [Walker *et al.*, 2002]) consist of human-machine dialogues (approx 2300 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 af-

ter the system thinks it has filled the relevant slots. These corpora have been previously annotated using the DATE scheme, for each *system* utterance's Conversational Domain, Speech Act, and Task [Walker *et al*., 2001b].

We used a hand-crafted automatic system to assign Speech Acts and Tasks to the user utterances, and to compute information states for each point in the dialogue (i.e. after every utterance). The system is implemented using DIPPER [Bos *et al*., 2003] and OAA [Cheyer and Martin, 2001], using several OAA agents (see [Georgila *et al*., 2005] for more details). An example of some of the types of information recorded in an information state is shown in figure 1. The state is intended to record all the information about the preceding portion of the dialogue which is relevant to making dialogue management decisions, including filled slots, confirmed (i.e. grounded) slots, and previous speech acts.

For the experiments reported in this paper, we used a preliminary version of the annotation, which included 4 of the 8 systems in the 2001 corpus. This subset consists of 97 users, 697 dialogues, and 51,309 total states. The fact that the annotation was done automatically means that some errors are inevitable, particularly in this preliminary version. But we believe that this has an equal effect on our performance measures for both the COMMUNICATOR systems and our learnt systems, so it does not affect our conclusions.

3 Using the Data for Reinforcement Learning

We use the annotated COMMUNICATOR data to train a Reinforcement Learning system. In RL, the objective of the system is to maximise the reward it gets during the course of the dialogue. Rewards are defined to reflect how well a dialogue went, so by maximising the total reward the system optimises 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 before when 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 (the Q-function). Given a state and an action that could be taken in that state,[1] the Q-function tells us what total reward, on average, we can expect between taking that action and the end of the dialogue. Once we have this function, the optimal dialogue management policy reduces to simply choosing the action which maximises the expected future reward for the current state.

The actions which the reinforcement learning system needs to choose between are defined in terms of the DATE scheme [Walker and Passonneau, 2001] system annotations for Conversational Domain, Speech Act and Task. Each possible triple of values for these three features is considered a different action. In addition, there are release_turn and end_dialogue actions. There are a total of 70 actions which occur in the data.

[1]The expected future reward also depends on the dialogue management policy which 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 below.

3.1 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 extracted from user questionnaires and task completion measures. For all states other than the final state, we provide a reward of -1. 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 which 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, and 0 otherwise. The remaining reward features have values ranging from 1 to 5 in the annotation. Their reward is their value (minus one) times the weight shown in table 1. The relative values of these later weights was determined by the empirical analysis reported in [Walker *et al*., 2001a].

Actual Task Completion	100
Perceived Task Completion	100
Task Ease	9
Comprehension Ease	7
System behaved as Expected	8
Future Use	9

Table 1: The weights used to compute a dialogue's final reward value. The first two features' weights are multiplied by 0 or 1, and the rest are multiplied by values from 0 to 4.

3.2 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 ("$E[\,]$") of the total reward between taking action a in state s_i until the end of the dialogue.

$$Q(s_i, a) \approx E[\sum_{j>i} r(d,j) | s_i, a]$$

Given that the number of possible future state sequences $(s_{i+1}, \ldots)$ 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 number of possible states is extremely large. Having a large number of states is a more realistic scenario for a 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 each 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

```
STATE 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]
FilledSlotValue: [late morning]
FilledSlot: [depart_time]
CommonGround: [dest_city]

LOW LEVEL
WordErrorRate: 20.00

HISTORY LEVEL
SpeechActsHist: [], opening_closing,[], opening_closing, instruction, request_info,
  [provide_info], implicit_confirm, request_info, [provide_info], implicit_confirm,
  request_info, [provide_info]
TasksHist: [], 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: [], [], [orig_city], [dest_city], [depart_time]
FilledSlotsValuesHist: [], [], [hartford connecticut], [orlando florida], [late morning]
GroundedSlotsHist: [], [], [], [orig_city], [dest_city]
```

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

of techniques that could be used for RL with feature-based representations of states, but the simplest and most efficient is linear function approximation.

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 $Q(s,a)$ 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 dataset, 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].

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

The weights w_a are learnt from data, but the mapping $f(s)$ from states to vectors must be specified beforehand. Each value $f_i(s)$ in these vectors represents a possible commonality between states, so it is through the definition of $f(s)$ that we control the notion of commonality which will be used by the linear function approximation. The definition of $f(s)$ we are currently using is a straightforward mapping from feature-value pairs in the information state s to values in the vector $f(s)$.

The state vector mapping $f(s)$ is computed using the first four levels of our annotation of the COMMUNICATOR data. We went through these annotations and identified the features which we consider relevant for dialogue management. These features were of three types. For features which 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. For features which can have arbitrary text as their values, we used 1 to represent the presence of text and 0 to represent no value. The remaining features all have either a finite set of possible values, or a list of such values. Features with a list value are first converted to a list of pairs consisting of the feature and each value. For every possible feature-value pair, we define an element of the vector $f(s)$ which is 1 if that feature-value pair is present in the state and 0 if it is not. These form the vast majority of our features. In total there are 291 features.

To train the weights of the linear approximation, we employed a standard RL learning method called SARSA(λ) [Sutton and Barto, 1998]. 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 bring us into the current research topic of kernel-based methods. Kernels are used to compensate for the over-simplicity of linear functions, and can be used to express more complicated notions of commonality between states [Shawe-Taylor and Cristianini, 2004].

3.3 Applying RL to a Fixed Dataset

We initially tried using the estimate of expected future reward $Q_{data}(s,a)$ discussed in the previous section to define our dialogue policy. The dialogue policy simply selected the action a with the highest $Q_{data}(s,a)$ given the state s. However, we found that this policy was very different from the policy ob-

[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").

served in the COMMUNICATOR data, almost never choosing the same action as was in the data. This simply means that the actions which have been learnt to have the best future rewards are not the ones that were typically chosen by the COMMUNICATOR systems in those states. Such actions would then lead to states unlike anything observed in the data, making the estimates for these states highly unreliable. In addition, the future reward depends on the policy the system uses in the future, so if the policy is different from that observed in the data, then the estimate $Q_{data}(s,a)$ is not even relevant.

The solution to these problems which is typically used in RL research is to generate new data as learning progresses and the policy changes. The RL system can thus explore the space of possible policies and states, generating new data which is relevant to each explored policy and its states. Such policy exploration is often considered an integral part of RL. In future research, we intend to perform this exploration by running each policy with a user simulation (as in [Scheffler and Young, 2002]) trained on the COMMUNICATOR dataset, but first we need a solution to the problem of applying RL to a fixed set of data. Such policy exploration is only feasible with simulated dialogues generated through interaction with a simulated user, because generating real data with human users is very expensive. But a simulated user is not the same as a human user, so it is important to learn as much as possible from the fixed set of data we have from human users. In addition, the huge policy space makes even policy exploration with simulated users intractable, unless we can initialise the system with a good policy and constrain the exploration. This also requires learning as much as possible from the fixed set of data available before exploration.

There have been some proposals for learning a policy which 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 which has been observed in the data.

3.4 A Hybrid Approach to RL

To address the above problems, we have investigated a novel hybrid approach which combines RL with supervised learning. The supervised learning is used to model the policy which the systems in the data actually use, which we model as a probabilistic policy $S_{data}(s,a)$.

$$S_{data}(s,a) \approx P(a|s)$$

In other words, $S_{data}(s,a)$ is an estimate of the probability that a random system selected from those which generated the data would choose action a given that it is in state s. The function $S_{data}(s,a)$ is computed with linear function approximation, just like $Q_{data}(s,a)$, except that a normalised exponential function is used so that the result is a probability distribution over actions a.

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

As with the Q-function, the use of linear function approximation means that we have estimates for $P(a|s)$ even for states s which have never occurred in the data, based on similar states which did occur.

The hybrid approach we have investigated is based on the assumption that we can't model the expected future reward for states in the unobserved portion of the state space. Thus we simply specify a fixed reward for these unobserved 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 expected future reward is then the average between the fixed reward 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 fixed reward U for unobserved states with the $Q_{data}(s,a)$ estimate for observed states, where the mixture coefficient is the probability $P_{observed}(s,a)$ that performing a in s will lead to an observed state.

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

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 which discriminates between different actions in the same way as this estimate. To derive such a discriminant function, we first approximate $P_{observed}(s,a)$ in terms of the probability distribution in the data $P(s,a)$ and the size of the dataset N, under the assumption that the number of possible state-action pairs is much larger than the size of the dataset (so $P(s,a)N \ll 1$).

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

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

$$\begin{aligned} &Q_{data}(s,a_1)S_{data}(s,a_1)P(s)N + U(1 - S_{data}(s,a_1)P(s)N) \\ &\leq Q_{data}(s,a_2)S_{data}(s,a_2)P(s)N + U(1 - S_{data}(s,a_2)P(s)N) \end{aligned}$$

if and only if

$$S_{data}(s,a_1)(Q_{data}(s,a_1) - U) \leq S_{data}(s,a_2)(Q_{data}(s,a_2) - U)$$

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

$$Q_{hybrid}(s,a) = S_{data}(s,a)(Q_{data}(s,a) - U)$$

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 maximising $Q_{hybrid}(s,a)$ is equivalent to maximising $S_{data}(s,a)$.

4 Experimental Results

We evaluate the trained dialogue management policies by running them against trained user simulations. Both the policies and the user simulations were trained using the annotated COMMUNICATOR data for the ATT, BBN, CMU, and SRI

systems. We compare our results against the performance of these same four systems, using an evaluation metric discussed below. The information states for the simulated dialogues were computed with the same rules used to compute the information states for the annotated data.

4.1 The Testing Setup

For these experiments, we restrict our attention to users who only want single-leg flight bookings. This means there are only 4 essential slots to be filled: origin city, destination city, departure date, and departure time. To achieve this restriction, we first selected all those COMMUNICATOR dialogues which did not contain trip continuations.[3] This subset contained 79 BBN dialogues, 132 CMU dialogues, 258 ATT dialogues, and 174 SRI dialogues. This subset was used for evaluating the systems and for training the user model. The system model was trained on the full set of dialogues, since it should not know the user's goals in advance.

The user model was trained in the same way as the supervised component of the hybrid system discussed above, using linear function approximation and a normalised exponential output function. The states which precede user actions are input as vectors of features very similar to those used for the system but tailored to the needs of a user model. The output of the model is a probability distribution over actions, which consist of Speech Act, Task pairs. 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.

When we first tested the hybrid policy, we found that it never closed the dialogue. We think that this is 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, which we plan to address by splitting this action into two actions, one for "greeting" and one for "goodbye". In the meantime, we have augmented the hybrid policy with a rule which closes the dialogue after the system chooses the action "offer", to offer the user a flight. We have also added rules which close the dialogue after 100 states (i.e. total of user and system actions), and which release the turn if the system has done 10 actions in a row without releasing the turn.

4.2 The Evaluation Metric

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, 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 flight bookings, the scoring function only looks at the four slots relevant to these bookings: origin city, destination city, departure date, and departure time. We give 25 points for each slot which is filled, plus another 25 points for each slot which is also confirmed (i.e. grounded). We also deduct 1 point for each action performed by the system, to penalise longer dialogues. Thus the maximum possible score is 198 (i.e. 200 minus 2 system actions: ask for all the user information in one turn, and then offer a flight).

The motivation behind this evaluation metric is that confirmed slots are more likely to be correct than slots which 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. 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 system value partial completions more highly, but inspection of the distributions of scores indicates that this difference does not favour 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 different systems. First, the exact same metric is applied to every system. Additional information which we have for some systems, but not all, such as the COMMUNICATOR user questionnaires, is not used. Second, the systems are being run against approximately equivalent users. The user simulation is trained on exactly the same user actions which 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 a speech recognition errors, since 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 favour of one system or another.

4.3 Comparisons Between Systems

We have run experiments to answer two questions. First, in our hybrid policy, what is the best balance between the supervised policy and the reinforcement learning policy? Second, how well does the hybrid policy perform compared to the COMMUNICATOR systems that it was trained on?

We trained models of both $Q_{data}(s,a)$ and $S_{data}(s,a)$, and then used them in hybrid policies with various values for the unobserved state penalty U. For both functions, we trained them for 100 iterations through the training data, at which point there was little change in the training error. During testing, each hybrid policy was run for 1000 dialogues against the linear function approximation user model. The final state

[3] There are only 54 dialogues which contain continuations. Excluding these dialogues does not harm the evaluation of the COMMUNICATOR systems, since their average score is actually lower than that for the non-continuation dialogues (20.9).

U	total score	filled slots	grounded slots	length penalty
-1000	114.2	89.1	47.4	-22.2
0	101.3	69.5	51.6	-19.7
40	100.6	69.9	51.9	-21.1
80	96.0	67.0	49.4	-20.4
vs Ngram:				
80	105.7	68.8	57.2	-20.3

Table 2: The average scores for different values of the unobservable state reward U, and the three components of these scores.

for each one of these dialogues was then fed through the scoring function and averaged across dialogues. The results are shown in table 2. The values for U were chosen based on the average number of decisions per dialogue which were different from that which the purely supervised policy would pick, which were 0 ($U = -1000$), 1 ($U = 0$), 2 ($U = 40$), and 5 ($U = 80$), respectively.

Table 2 also shows some results for running the hybrid system against a user simulation based on n-grams of actions ("vs Ngram"). This user model seems to be easier to interact with than the linear user model. In particular, the resulting dialogues are better in terms of grounding.

To evaluate how well the hybrid policy performs compared to the COMMUNICATOR systems, we extracted the final states from all the non-continuation dialogues and fed them through the scoring function. The average scores are shown in tables 3 and 4, along with the best performing hybrid policy and the scores averaged over all systems' dialogues.

Table 3 shows the results computed from the complete dialogues. These results show a clear advantage for the hybrid policy over the COMMUNICATOR systems. The hybrid policy fills more slots and does it in 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 the ordering of the systems is the same. They do better on length, but worse on grounding, and about the same on filled slots.

4.4 Discussion of Results

These results provide clear answers to both our questions, for this corpus and this approach to dialogue management. First, the more the system relies on the policy determined with supervised learning, the better it does. Second, the learnt policies perform better than any of the COMMUNICATOR systems.

The results in table 2 show a clear trend whereby the more the system sticks with the supervised policy, the better it does. In other words, the best the hybrid policy can do is simply mimic the typical behaviour it observes in the systems in the data. This is a surprising result, in that we were expecting reinforcement learning to provide some improvement over the supervised policy, provided the hybrid policy wasn't too different from the supervised policy. This may reflect the fact that RL is harder than supervised learning, or that the amount of data we are using isn't enough to train an RL system effectively for such a complicated dialogue management task. We are currently producing better quality annotations, for the full set of COMMUNICATOR systems, which should improve the estimates for the RL component of the hybrid system. Another approach would be to only use the RL component for specific types of decisions (thereby simplifying the dialogue management task for RL). For example, the results in table 2 suggest that reinforcement learning improves grounding, but perhaps ends the dialogue before all the slots are filled.

System	total score	filled slots	grounded slots	length penalty
hybrid RL/SL	114.2	89.1	47.4	-22.2
BBN	67.5	77.2	59.2	-68.9
CMU	49.7	60.2	47.4	-57.9
ATT	39.5	55.6	33.1	-49.3
SRI	21.0	52.4	0.0	-31.4
combined COMM	40.0	58.4	30.3	-48.6

Table 3: The average scores for the different systems, and the three components of these scores.

System	total score	filled slots	grounded slots	length penalty
hybrid RL/SL	114.2	89.1	47.4	-22.2
BBN	83.2	74.1	24.1	-15.0
CMU	63.9	55.1	26.9	-18.1
ATT	55.3	55.8	24.4	-25.0
SRI	27.8	52.2	0.0	-24.4
combined COMM	53.0	56.9	18.3	-22.2

Table 4: The average scores after the first flight offer for the different systems, and the three components of these scores.

It is worth noting that, even though the best system is purely supervised, the hybrid policies which do use some RL also do quite well. This can be seen by comparing their scores to the results for the COMMUNICATOR systems in tables 3 and 4. This performance is achieved despite the extreme difficulty of our state and policy spaces, as indicated by the very bad performance we observed when we initially tried a purely RL system. Therefore, we conclude that this hybrid approach would allow the fruitful use of RL in many situations where RL would otherwise be inappropriate due to the complexity of the task or the amount of data available. We anticipate that even for our very complex task, RL can be made beneficial by increasing the amount of data through simulated dialogues, which we intend to do in future work.

Tables 3 and 4 show a clear advantage of the learnt policy "hybrid RL/SL" over all the COMMUNICATOR systems. This result is perhaps surprising, since the learnt policy shown is the purely supervised version, which simply mimics the typical behaviour of these same systems. One likely explanation is that the hybrid policy represents a kind of multi-version system, where decisions are made based on what the majority

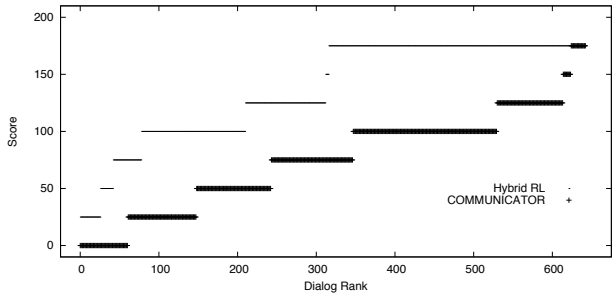

Figure 2: Comparing slot scores for the TALK learnt policy ("Hybrid RL") versus the COMMUNICATOR systems (disregarding penalties for dialogue length)

of systems would do. Multi-version systems are well known to perform better than any one system alone, because the mistakes tend to be different across the different systems.

For a more detailed comparison of the systems, figure 2 plots the scores (ignoring length) for the average COMMUNICATOR dialogues versus the learnt policy's simulated dialogues, as a function of their rank. Because the COMMUNICATOR systems do better when stopped after the first flight offer, we use these results for these plots. In figure 2, the length of each bar reflects how many dialogues achieved the associated score for the number of filled and/or grounded slots. None of the dialogues actually get the maximum score of 200 (all slots filled and grounded), but very many of the learnt policy dialogues score 175, compared to very few of the COMMUNICATOR dialogues reaching that score.

5 Conclusion

The learnt policy scores 37% higher than the best COMMUNICATOR system which we examined (114.2 versus 83.2). These are extremely promising results for a learnt policy with 70 actions, over 10^{87} possible states, and very few hand-coded policy decisions. They indicate that linear function approximation is a viable approach to the very large state spaces produced by the ISU framework. They also show that this method for combining supervised learning with reinforcement learning is effective at learning policies in extremely large policy spaces, even with the limited amount of data in any fixed dataset. In the case of only using supervised learning, the linear function approximation is able to merge the policies of the systems in the COMMUNICATOR data and perform better than any one of these systems alone. Currently, adding reinforcement learning to this model degrades performance slightly, but still does better than any of the COMMUNICATOR systems. Further improvement should be possible by tailoring the representation of states and actions based on our experience so far.

The next step is to better exploit the advantages of reinforcement learning. One promising approach is to apply RL while running the learnt policy against simulated users, thereby allowing RL to explore parts of the policy and state spaces which are not included in the COMMUNICATOR data. The hybrid policy we have learnt 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 which need to be explored when training the RL component. All of these advances will improve techniques for bootstrapping and automatic optimisation of dialogue management policies from limited initial datasets.

Acknowledgements

This work is funded by the EC under the FP6 project "TALK: Talk and Look, Tools for Ambient Linguistic Knowledge". We thank Johanna Moore for proposing the use of the COMMUNICATOR dataset for this work.

References

[Bos *et al.*, 2003] Johan Bos, Ewan Klein, Oliver Lemon, and Tetsushi Oka. DIPPER: Description and Formalisation of an Information-State Update Dialogue System Architecture. In *4th SIGdial Workshop on Discourse and Dialogue*, pages 115–124, Sapporo, 2003.

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

[Georgila *et al.*, 2005] Kallirroi Georgila, Oliver Lemon, and James Henderson. Automatic annotation of COMMUNICATOR dialogue data for learning dialogue strategies and user simulations. In *Ninth Workshop on the Semantics and Pragmatics of Dialogue (SEMDIAL), DIALOR'05*, 2005.

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

[Levin and Pieraccini, 1997] E. Levin and R. Pieraccini. A stochastic model of computer-human interaction for learning dialogue strategies. In *Proceedings of Eurospeech*, pages 1883–1886, Rhodes, Greece, 1997.

[Levin *et al.*, 2000] E. Levin, R. Pieraccini, and W. Eckert. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, 8(1):11–23, 2000.

[Scheffler and Young, 2002] Konrad Scheffler and Steve Young. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proc. HLT*, 2002.

[Shawe-Taylor and Cristianini, 2004] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[Singh *et al.*, 2002] Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research (JAIR)*, 2002.

[Sutton and Barto, 1998] Richard Sutton and Andrew Barto. *Reinforcement Learning*. MIT Press, 1998.

[Walker and Passonneau, 2001] M. Walker and R. Passonneau. DATE: A Dialogue Act Tagging Scheme for Evaluation of Spoken Dialogue Systems. In *Walker, M., Passonneau R., DATE: A Dialogue Act Tagging Scheme for Evaluation of Spoken Dialogue Systems. In Proceedings of Human Language Technology Conference, San Diego, March, 2001.*, 2001.

[Walker *et al.*, 2001a] M Walker, 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. Darpa communicator dialog travel planning systems: The june 2000 data collection. In *Eurospeech 2001*, Aalborg, Scandinavia, 2001.

[Walker *et al.*, 2001b] Marilyn A. Walker, Rebecca J. Passonneau, and Julie E. Boland. Quantitative and Qualitative Evaluation of Darpa Communicator Spoken Dialogue Systems. In *Meeting of the Association for Computational Linguistics*, pages 515–522, 2001.

[Walker *et al.*, 2002] M. Walker, 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. Darpa communicator: Cross-system results for the 2001 evaluation. In *ICSLP 2002*, 2002.