The Markov Assumption in Spoken Dialogue Management

Tim Paek & David Maxwell Chickering
One Microsoft Way
Redmond, WA 98052 USA
{timpaek|dmax}@microsoft.com

Abstract

The goal of dialogue management in a spoken dialogue system is to take actions based on observations and inferred beliefs. To ensure that the actions optimize the performance or robustness of the system, researchers have turned to reinforcement learning methods to learn policies for action selection. To derive an optimal policy from data, the dynamics of the system is often represented as a Markov Decision Process (MDP), which assumes that the state of the dialogue depends only on the previous state and action. In this paper, we investigate whether constraining the state space by the Markov assumption, especially when the structure of the state space may be unknown, truly affords the highest reward. In a simulation experiment conducted in the context of a dialogue system for interacting with a speech-enabled web browser, models under the Markov assumption did not perform as well as an alternative model which attempts to classify the total reward with accumulating features. We discuss the implications of the study as well as limitations.

1 Introduction

The goal of dialogue management in a spoken dialogue system is to take actions based on observations and inferred beliefs. Dialogue management plays a crucial role in the overall performance of the system since speech recognition is often quite poor, due to noisy or unexpected input. With robust dialogue management, the system can still take actions that maintain the task at hand. Unfortunately, coming up with a suitable set of dialogue management strategies is no easy task. Traditional methods typically involve authoring and tuning complicated hand-crafted rules that require considerable deployment time and cost. Statistical methods, on the other hand, hold the promise of robust performance from models that can be trained on data and optimized, so long as the data is representative of what the dialogue system can expect to encounter during deployment (Young, 2000).

Among the more popular statistical methods, researchers have turned to reinforcement learning methods since it is possible to derive a policy for action selection that is guaranteed to be optimal with respect to the data given that the dynamics of the system is represented as a Markov Decision Process (MDP), which assumes that the state of the dialogue depends only on the previous state and action. The Markov assumption is made as a modeling choice for the data. Hence, an important topic of inquiry is whether this choice is appropriate and beneficial.

In this paper, we explore the Markov assumption on both theoretical and empirical grounds. In particular, we investigate whether constraining the state space by the Markov assumption truly affords the highest reward, especially when the structure of the state space may be unknown, which is typically the case. This paper is organized as follows. In Section 2, we provide relevant background on reinforcement learning with specific focus on the modeling assumptions relevant to spoken dialogue. In Section 3, we challenge the modeling assumptions by proposing alternative models to the MDP that vary the temporal relations among features. All competing models generate dialogue management strategies for interacting with a speech-enabled web browser, and we explain in detail how we built these models from data. In Section 4, we evaluate the performance of all the models in a simulation experiment. Finally, in Section 5, we discuss the implications and limitations of the experimental study.

2 Background

Reinforcement learning addresses the problem of how an agent should act in dynamic environments so as to maximize a scalar reward signal (Sutton & Barto, 1998).

This problem is manifest in spoken dialogue systems since the system must take sequential actions based on its observations, such as user utterances, and its beliefs. A central debate in the literature concerns the use of models. *Model-free* approaches do not explicitly represent the dynamics of the environment, but instead directly approximate a value function that measures the desirability of each environment state. These approaches offer near-optimal solutions that depend on systematic exploration of all actions in all states (Watkins & Dayan, 1992). On the other, *model-based* approaches explicitly represent a model of the dynamics of the environment to compute an estimate of the expected value of each action. With a model, the agent can reduce the number of steps to learn a policy by simulating the effects of its actions at various states (Sutton & Barto, 1998). Perhaps for this reason, and for the fact that it is possible to derive a policy that is guaranteed to be optimal with respect to the data, spoken dialogue researchers have by and large pursued model-based reinforcement learning methods (see e.g., Levin et al., 1998; Singh et al., 2002).

The framework underlying model-based reinforcement learning is that of the MDP, which can be characterized by a tuple (*S*, *A*, *P*, *R*) with:

- A state space *S* with states $s \in S$. The state space may consist of features related to spoken utterances, user and system actions, and so forth. We discuss this further in the next section.
- An action space A with actions $a \in A$. The action space comprises all system actions in dialogue management, such as confirming various slots, or engaging in a user requested service.
- Unknown state transition probabilities $P : S \times A \times S \rightarrow [0,1]$, where $P(S_{t+1} \mid S_t, A_t)$ gives the probability of a transition from a state $s \in S$ and action $a \in A$ at time slice *t* to another state $s \in S$ in the next time slice. The distribution *P* defines the dynamics of the environment, and constitutes the basis for the Markov assumption.
- A reward function $R : S \times A \rightarrow \Re$, where $R_t = R(S_t, A_t)$ assigns an immediate reward at time slice *t* for taking action $a \in A$ in state $s \in S$. *R* plays a critical role in the policy that is learned for dialogue management as we discuss further below.

In order for a dialogue system to take actions according to the MDP, it is necessary to be able to derive a policy $\pi : S \rightarrow A$ mapping states to actions so that it maximizes some specified objective function for the long term reward. How much of the future the system takes into account in making its decisions at any given moment depends upon the specified horizon for the objective function. Perhaps the simplest objective function is the total reward over a finite horizon, which specifies that at any given time *t*, the system should optimize its expected reward for the next *h* steps:

$$E\left(\sum_{q=t}^{t+h} R_q\right) \tag{1}$$

Alternatively, the system can take the infinite long term reward into account with future rewards geometrically discounted by a discount factor γ so as to motivate the dialogue system to complete the interaction as soon as possible:

$$E\left(\sum_{q=t}^{\infty} \gamma^q R_q\right) \tag{2}$$

where $0 \leq \gamma < 1$ for computability purposes. While it is theoretically possible for a spoken dialogue to continue ad infinitum, most systems are designed to avoid infinite regressions where, for example, the system engages in the same repair over and over (e.g., "I'm sorry, can you repeat that?"). In practice, most dialogues are finite horizon, given that oftentimes growing user frustration ultimately leads to the termination of the interaction.

In place of (1) and (2) above, the objective function can also be based on post-hoc measures such as usability scores (Singh et al., 2002; Walker et al., 2001b), and construed to reflect whatever qualities a dialogue designer may want the system to possess, such as the ability to re-tool the system for future use (Walker et al., 2001a). In short, the assignment of the reward function reflects the desired behavior of the system.

For the rest of this paper, we confine our discussion to the finite horizon MDP where we assume for simplicity that all variables in the state space can be fully observed by the system. When state variables are included that are not fully observable, such as the user's intention in producing an utterance, the dialogue constitutes a Partially Observable MDP (see e.g., Paek & Horvitz, 2000; Roy et al., 2000; Zhang et al., 2001). The POMDP also employs the Markov assumption.

Building on (1), an optimal policy can be learned through various algorithms that involve finding the optimal value function:

$$V_t^*(s_t) = \max_{\pi} E\left(\sum_{q=t}^{t+h} R_q\right) \tag{3}$$

where the optimal value of a state *s* is the expected reward for the next *h* steps, if the system starts in *s* at time

t and executes the optimal policy π. The optimal value function (3) is unique and can be defined recursively using the Bellman equations:

$$V_t^*(s_t) = \max_{a_t}\left(R_t + \sum_{s_t \in S} P(s_{t+1} \mid s_t, a_t) V_{t+1}^*(s_{t+1}) \right) \quad (4)$$

where the value of a state s at time t is the expected immediate reward plus the expected value of the next state at time $t+1$ using the best possible action. The simultaneous equations engendered by (4) can be solved efficiently with dynamic programming. Given the optimal value function, the optimal policy is simply:

$$\pi_t^*(s_t) = \arg\max\left(R_t + \sum_{s_t \in S} P(s_{t+1} \mid s_t, a_t) V_{t+1}^*(s_{t+1}) \right) \quad (5)$$

2.1 Influence Diagram

The finite horizon MDP can be viewed as a special case of an influence diagram, a more general framework for graphical modeling that facilitates decision-theoretic optimization. An influence diagram is a directed acyclic graph composed of three types of nodes: chance nodes, decision nodes and value nodes. The influence diagram also contains a single utility node that is a deterministic function of all the value nodes. Connecting the nodes are two types of arcs: probabilistic arcs and informational arcs. Arcs pointing into chance or value nodes specify a probabilistic dependency between a child and its parents. Arcs pointing into a decision node are "informational" in that the parents of the decision node are assumed to be known or observed before a decision is made. Although the traditional definition of an influence diagram (Howard & Matheson, 1981) permits only one value or utility node, our use of multiple value nodes is simply a way of factoring the utility function and has been used by other researchers (Tatman & Shachter, 1990; Lauritzen & Nilsson, 2001).

Figure 1 displays an influence diagram for a finite horizon MDP where all states $s \in S$ have been mapped to chance nodes, all actions $a \in A$ to decision nodes, and all R to value nodes expressing the immediate reward for taking action a in state s at time t. A utility node at the bottom of the Figure sums all the immediate rewards as in (3). Technically, since the MDP is fully observable at any given time slice, informational arcs point into each decision node from the previous time slice, though we have left them out to reduce clutter. The influence diagram also contains a set of parameters Θ that characterize the conditional distributions of the non-decision nodes, defined as:

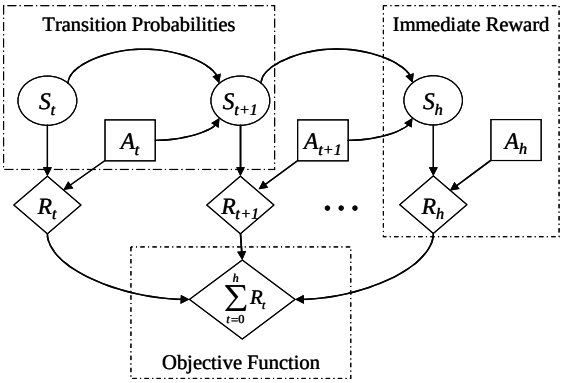

Figure 1. An influence diagram model of a finite horizon MDP. Informational arcs have been left out.

$$P(S, R \mid A, \Theta) = \prod_{X \in S \cup R} P(X \mid Pa(X), \Theta_X) \quad (6)$$

where $Pa(X)$ denotes the set of parents for node X, and Θ_x denotes the subset of parameters in Θ that define the local probability distribution of X. The transition probabilities P for the MDP are clearly subsumed by (6) and reside in the node S_{t+1}, as shown in the Figure.

We discuss influence diagrams for two reasons. First, influence diagrams allow us to understand what kinds of alternative models we could experiment with in competition to the MDP, since the transition probabilities P could easily depend on state variables other than just those in the previous time, as we demonstrate in the next section. And second, influence diagrams provide a framework in which to address what state variables should be included in S at all if a dialogue designer is unsure about what variables may be important for receiving an immediate reward.

2.2 MDP Assumptions

Before discussing the assumptions underlying the MDP, it is important to consider the basic units of dialogue modeling; that is, what basic units form a dialogue process. Since all dialogue systems respond to user utterances, perhaps the simplest way to model the dynamics of the interaction is to divide the temporal process by user utterances. In other words, a dialogue "turn" begins at the start of each new user utterance. While alternative ways of measuring time exist, such as question-answer pairs or contributions (Clark, 1996), they typically require knowledge about the type of utterance or action that was produced; for example, that an utterance was an "uptake." For simplicity, we take the user utterance as most basic unit of dialogue progression. Given an utterance then, the most basic features that a system can observe before taking an action are those that pertain to the utterance itself. As such, we consider that at every turn, a dialogue system can observe *at least*

the features that can be known about the current utterance at hand. We now discuss modeling assumptions that can be made on top of this basis.

The MDP framework relies on several assumptions, not all of which may be valid in the context of spoken dialogue. The most obvious assumption is the Markov assumption. One reason for making the Markov assumption is that it allows the Bellman equations in (4) to exploit the "*Optimality Principle*," which states that whatever the initial state may be, all remaining decisions must be optimal with regard to the state following the first decision (Bellman, 1957). This allows the optimal policy (5) to be solved efficiently using dynamic programming. However practical this reason may be, whether or not a model constrained by the Markov assumption yields the highest reward as compared to models constrained by other assumptions is still an empirical question, one which we investigate later.

From a linguistic perspective, it seems counter-intuitive to believe, as the Optimality Principle implies, that an optimal policy based just on the previous turn (i.e., the features of the previous utterance) provide as good of a policy as that based on the full history of interaction. After all, most linguists acknowledge that in a conversation, participants collaboratively build up shared knowledge about what has been said and mutually understood (Clark, 1996). This shared knowledge, or common ground, is cumulative in nature and underlies all future interactions. In the next section, we consider a cumulative model with no temporal assumptions in contrast to the MDP.

A response to this criticism is to argue that if aspects of history are important for making future decisions, they could be incorporated with global states that summarize what has been learned so far. However, this argument merely avoids the problem by adding additional assumptions, this time relating to what variables should be included in the state space. Most policy guided dialogue systems specify the state space up front, delineating all state variables that are assumed to be relevant for receiving a reward. These variables are defined and restricted so as to not only facilitate the Markov assumption, but also expedite tractable inference. Unfortunately, in practice, most of the time dialogue designers do not know in advance what variables should be included in the state space. In the next section, we discuss what a dialogue designer could do in such a situation. For now, it is enough to say that if possible, we should like to build models that rely on as few assumptions as necessary.

Finally, another assumption underlying the MDP is that the probabilities of making state transitions or receiving specific rewards do not change over time; that is, they are "stationary." For dialogue systems that provide services across a large population of users, the stationary assumption may indeed hold since individual differences are generalized. However, for dialogue systems that provide services to a limited number of users, it is not unreasonable to believe that people may change their preferences about how they want the system to behave around them over time. If unobservable states such as user frustration are included in the model, they may change over time as well. In such cases, it is incumbent upon the system to continually adapt its policy. Elsewhere, we discuss how a dialogue system could adapt its policy in real-time to a particular user through online feedback (Chickering & Paek, 2005).

3 Alternative Models

In the previous section, we discussed how the Markov assumption can be tied together with the selection of state space. Unfortunately, dialogue designers who want to utilize reinforcement learning for dialogue management typically do not know in advance what variables are relevant for receiving a reward and how they are related to each other: that is, the structure of the state space is unknown. Rather than choosing variables so as to facilitate the Markov assumption, we propose that the state space be learned from data along with the policy. This can be done using techniques for learning the parameters and structure of a Bayesian network, extended to influence diagrams (Heckerman, 1995; Chickering & Paek, 2005).

To derive the structure of the graphical models we describe below, including the MDP, we learned influence diagrams employing decision trees to encode local conditional distributions using a tool that performs Bayesian structure search (Chickering, 2002). Decision trees can be learned for both discrete and continuous variables, where splits in the trees are made through greedy search guided by a Bayesian scoring function (Chickering et al., 1997). In learning the influence diagrams, we only specified constraints on the state space, such as the Markov assumption, for which we wanted to conduct experiments. In particular, we built competing models to the MDP that vary the nature of the temporal relations between features. We did not assume that the state space was known beforehand, and as such, we included all variables that we were able to log for interacting with a command-and-control, speech-enabled web browser. We now describe the data collection and models we built.

3.1 Data Collection

The data from which we built all models was for spoken dialogue interaction with a speech-enabled web browser. As we describe in Chickering & Paek (2005), the data was generated using a simulation environment, where all possible system actions to a user command were systematically explored. The simulation pro-

ceeded as follows. First, we randomly selected a command from the command-and-control grammar for the browser (e.g., "go back", "go forward", "go to link x"). Using state-of-the-art TTS generation, we produced an utterance for the command, varying all possible TTS parameters, such as engine, pitch, rate, and volume. Since we were interested in building models that were robust to noise, we included empty commands and added various types of background noise to see if a model could learn to ignore spurious commands. The produced utterance was then recognized by a Microsoft Speech API (SAPI) recognition engine, whereupon we logged all possible SAPI events. These events, and functions of these events, constituted the feature set, which roughly fell into the following three broad categories:

1. *Within-utterance ASR features*: Features pertaining to a single utterance such as the number of hypotheses in an *n*-best list of variable length, the mean of the confidence scores, etc.
2. *Between-utterance ASR features*: Features pertaining to matches across utterances, such as whether the top rule in the *n*-best list matched the previous top rules, etc.
3. *Dialogue features*: Features pertaining to the overall dialogue such as the number of repairs so far, whether the system has engaged in a confirmation yet, etc.

It is important to note that both the between-utterance ASR features and dialogue features span multiple time slices. By including these features, we leave open the possibility that historical variables may very well be relevant for receiving a reward. In building the models, we let the learning algorithm decide whether to include these variables in its decision trees.

Once the produced utterance was recognized and events recorded, the simulation took a random action. For the first utterance, the simulation could either execute the most likely command in the *n*-best list (DoTop), confirm among the top three choices while giving the option that it may not be any of them (Confirm), ignore the utterance as spurious (Ignore), or ask for a repetition (Repeat). For the second utterance, Only DoTop, Confirm, and Repeat were possible, and for the third utterance, only DoTop and Bail, an action which makes an apology and terminates the dialogue, were possible. We did not allow the interaction to extend beyond the third utterance given the typically low tolerance users have in command-and-control settings for extended repairs.

If the simulation selected DoTop, Ignore, or Bail, the session finished and the simulation could now be rewarded. For taking the action DoTop, if the result of executing the most likely command matched the "true" command (i.e., the one sampled from the grammar), the simulation received a positive scalar reward (+100); otherwise, it received a negative reward (-100). For taking the action Ignore, if the true command was empty, it received +100, else -100. For Bail, it received -100. If either the repair action Confirm or Repeat was selected, a penalty (-75) was received, and the simulation proceeded to produce a confirmation choice that matched the true command or a repetition of the previous utterance respectively.

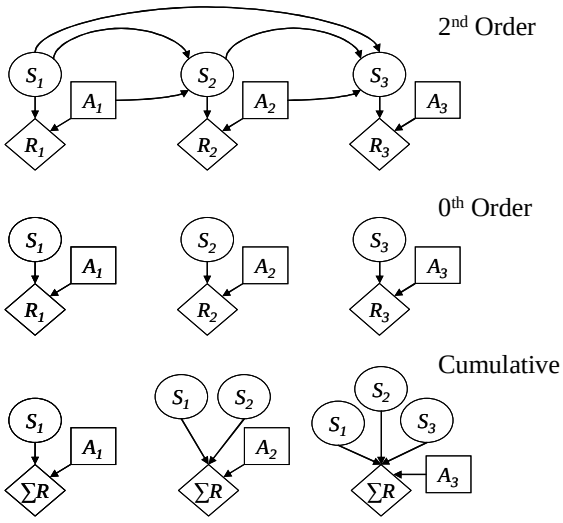

Figure 2. Alternative models to the MDP that vary along the Markov scale. The utility node in the top two models have been left out, as well as informational arcs in the second order model.

In summary, by sampling a command from the grammar and then responding to it with random actions until the dialogue finished and rewards were assigned, the simulation environment amassed data that could be used to learn which features and actions were most relevant for receiving a reward.

3.2 Types of Models

In order to validate empirically whether constraining the state space by the Markov assumption, especially when the structure of the state space may be unknown, truly affords the highest reward, we built three types of alternative influence diagrams that vary in the extent of their temporal dependencies using the simulation data.

Figure 2 displays the alternative models. Besides learning an MDP, which has a first order Markov dependency, we also learned a second order Markov model where the third time slice state variables could also depend on those in the first time slice, as shown in the top of the Figure. In preparing the data for the second order model, we added between-utterance ASR features such as where the top command in the third slice occurred in the *n*-best lists of the second and first

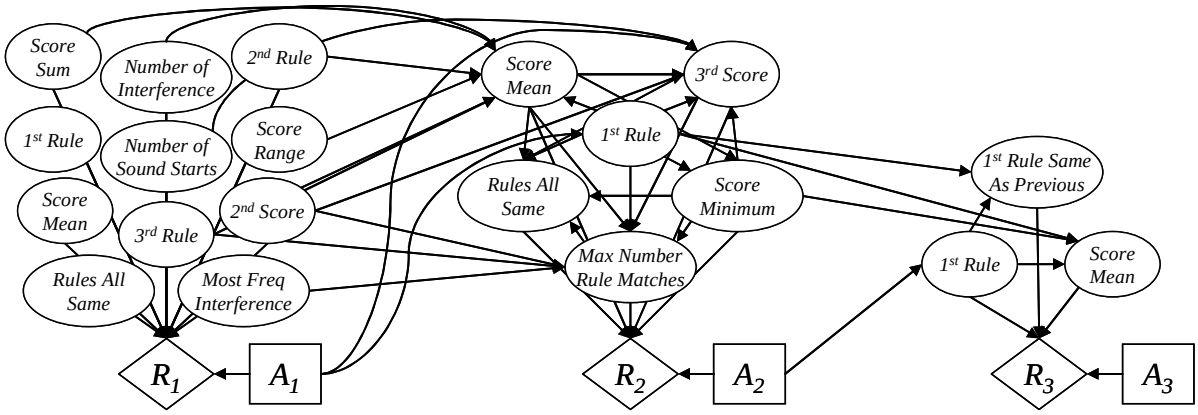

Figure 3. A finite horizon MDP learned from data, where the state space was constrained to include only those variables that directly predicted the immediate reward for each time slice, in addition to the Markov assumption. The overall value node has been left out, as well as the informational arcs.

time slices. We also built a zero order model without any dependencies between time slices, as shown in the middle of the Figure.

Finally, we built a cumulative total reward model for each time slice where state features accumulate with each successive time slice, as shown in the bottom of the Figure 2. For this model, the local system actions optimize the total utility function, as opposed to the immediate reward function. This kind of model had been applied previously in the call-routing domain to transfer calls so as to minimize caller time and operational expense (Paek & Horvitz, 2004). In effect, the cumulative total reward model falls under the rubric of a "model-free" approach in that it does not explicitly represent the dynamics of the environment, but rather directly predicts the long term reward. Fundamentally, the cumulative model attempts to classify the total reward, which given the small state space for our domain, was discrete, though in more complicated domains a regression could be learned.

Building each model entailed piecewise construction of portions of the model by the time slice variables involved. For those models involving just the first time slice variables, we had over 20,000 simulation sessions as training data. For those models involving the first and second time slices, we had over 10,000 sessions. And finally, for those models involving all three time slices, we had over 5,000 sessions.

Model	# Splits	# State Space Variables
0th Order	903	80
1st Order	1096	90
1st Order (limited)	183	20
2nd Order	1133	90
2nd Order (limited)	190	20
Cumulative (combined)	19	9

Table 1. Complexity of the models learned as a function of the number of splits in the decision trees and the number of state space variables.

Table 1 summarizes the complexity of the models as a function of the number of splits in the decision trees that make up the local probability distributions, and the number of state space variables. For the first and second order Markov models we also built "limited" versions, where the state space variables were limited to those that directly predicted the immediate reward as parents. This had the effect of reducing the complexity of each respective model by a factor of 6 for the number of splits, and 4 for the number of state variables. The simplest model by far was the cumulative total reward model which had roughly 50 times less splits than any of the unlimited Markov models, and 10 times less variables than the MDP model.

Figure 3 shows the limited MDP learned from the simulation data. For the first time slice, no dependencies between state variables are necessary since all features of the first utterance are fully observed. Note that each time slice has a number of features related to the *n*-best list confidence scores. This is not surprising given that most hand-crafted dialogue management strategies utilize some kind of confidence threshold for taking actions (e.g., "Do the top recognized command if its confidence is greater than 95%). The mean confidence score in the *n*-best list appears in every time slice suggesting that it pays off to build features that aggregate the confidence scores. In fact, many of the features in the first and second time slices of the limited MDP are aggregate features of the *n*-best list, such as the sum of all confidence scores ("Score Sum"), the range of score values ("Score Range"), and whether all the rules in the

list were the same though the actual phrases or wording were different ("Rules All Same").

The MDP in Figure 3 also shows that all time slices include a state variable related to the grammar rule that was observed, such as the first or top rule in the *n*-best list. This indicates that the model is domain-dependent; that is, the grammar rules specific to this application make a difference in whether or not the system receives a reward. For example, if we look at the decision tree for the value or reward node of the second time slice, we would see that if the top rule in the *n*-best list is "hide numbers" (i.e., numbered hyperlinks) after the system has asked for a repeat in the first time slice, and the system decides in this current time slice to execute this top rule, the probability that it would receive an immediate negative reward for failure would be 90%. All the models we learned from the data were domain-dependent in this way.

Some of the variables that were not included in the limited MDP, but were included in the unlimited version were most of the between-utterance ASR features. The notable exception is the variable "1st Rule Same As Previous," which checks to see if the top rule between the current time slice and previous time slice are the same. In general, most of the state variables included in the limited models were by and large within-utterance ASR features.

3.3 Policy estimation

To derive an optimal policy from data involves a two step process: first, learning an optimal model with respect to the data, and then, learning an optimal policy with respect to the model. As discussed previously, we learned the models from the simulation data using Bayesian structure learning. In order to derive the optimal policy, the straightforward approach is to exploit the Markov assumption, at least for the MDP, and apply dynamic programming in the usual way. However, computing the policy for a state space that includes more than a handful of variables can be quite challenging, even for the limited model in Figure 3. This has prompted researchers to investigate techniques for factoring the state space using graphical models as we have in our models (Meuleau et al., 1998). An alternative approach for computing the policy, which we utilized, is to use forward sampling to approximate the dynamic programming solution for the MDP (Kearns et al., 1999). Since all the models we learned are generative models, we can approximate the expectation in (4) by sampling *N* random states from the next time slice so that the optimal value function becomes:

$$V_t^*(s_t) = \max_{a_t}\left(R_t + \frac{1}{N}\sum_{s_t \in S} V_{t+1}^*(s_{t+1}) \right) \qquad (6)$$

This allows us to compute the action with the greatest expected reward at any given time slice in the same way for every Markov model, making it easier to compare and evaluate them. For the cumulative total reward model, since the model directly predicts the total reward, we do not need to use sampling for inference as it contains its own policy for each time slice in the decision tree for the utility node.

4 Evaluation

Since it is often difficult to know in advance what variables in the state space are relevant to receiving a reward, we learned both the structure of the state space and policy for all the models discussed in the previous section. The only factor that we varied was the set of constraints we put on the state space. In particular, we varied the Markov order from zero to three. We also either limited the model to just the state variables that predicted the immediate reward or not, and finally, we built a cumulative total reward model that contains its own policy. All other factors being equal, the question remains as to whether or not constraining the state space by the Markov assumption truly affords the highest reward.

To answer this question, we used the same simulation environment we described earlier to create the training data to also generate 1000 experimental trials. For any utterance, different models could take different actions, and as a consequence, receive different responses. To maintain controlled experimental conditions, we generated utterances and features for the entire tree of all possible action sequences. That way we ensured that every model received the exact same features if they took the same action, which also meant that between-utterance features utilized the same previous utterance features. To reduce variability, we also fixed the TTS to just one voice and one set of parameters. Furthermore, since all models were trained with different kinds of background noise, we also included the most challenging type of background noise (viz., people chattering in a cafeteria) in the experimental trials.

As a baseline measure, we included how any system would do if it followed the simple policy of always executing the top command in the *n*-best list. This baseline is meant to characterize the behavior of a system that follows no dialogue management strategies. It simply commits to the top recognized result and never engages in repair.

Figure 5 displays the average reward for all models compared to the baseline, and Figure 6 the total reward. Although seven lines should be represented, only four are visible because the first and second order Markov models, as well as their limited versions, performed exactly the same with respect to both average and total reward; that is, the 1st order, 1st order (limited), 2nd or-

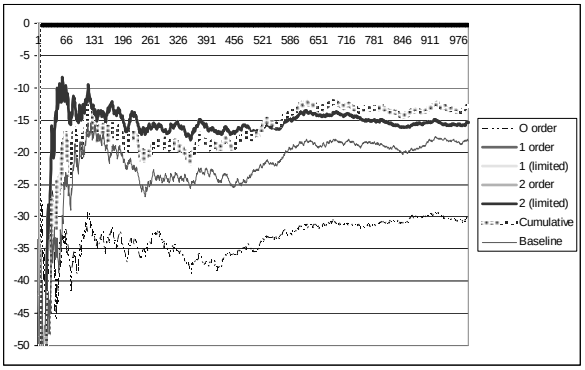

Figure 5. Average reward for all learned models over 1000 trials.

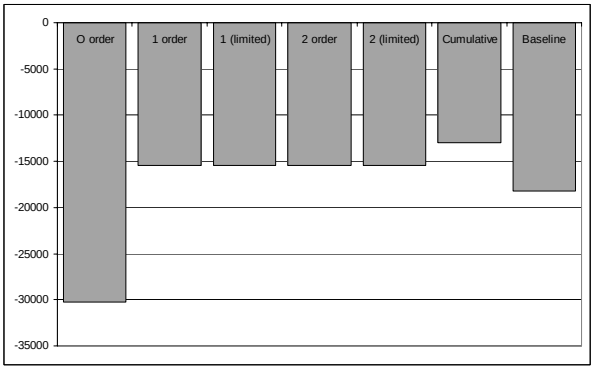

Figure 6. Total reward for all learned models after 1000 trials.

der, and 2nd order (limited) models were identical in performance. Hence, while the complexity of the models may have differed, they all took the same actions. This suggests that it is not worthwhile to constrain the state space beyond the first order Markov relation, nor is it worthwhile to include any more variables than those that directly predict the local immediate reward.

As far as the best performance is concerned, the first and second order models outperformed all other models for the first 500 trials, but then succumbed to the cumulative total reward model thereafter. By the end of the experiment, the cumulative total reward model had accrued the highest total reward, as shown in Figure 6. It is important to consider that the cumulative total reward model achieved this performance with only 9 state variables and 19 splits, combined between the three time slices.

Surprisingly, the zero order Markov model performed worse than the baseline. The reason for this dealt with the two repair actions, Repeat and Confirm. Without knowing what action was previously selected, since dependencies were not permitted between time slices, the model assigned the same probability of receiving an immediate negative or positive reward to both Repeat and Confirm, thereby conflating the two. To get around this problem, separate models specifically for Repeat and Confirm would need to be learned, though it is doubtful that this would have changed the final result of the experiment.

4.1 Assessing the Winner

Despite the relatively small complexity of the cumulative total reward model, it outperformed all other models. It is instructive to look at the state space variables that were included in the model for each time slice, as shown in Figure 7. For the most part, every state space variable in each time slice also appears in the limited MDP displayed in Figure 3. The only exception is "Number of Sound End Events," which is almost always the same in the data as the feature "Number of Sound Start Events" in the MDP. The primary difference between the cumulative model and the MDP then is that the former predicts the total reward as opposed to the immediate reward.

Ironically, although the cumulative model could have included previous state space variables in the second and third time slices, it did not. Instead, the most important variables for predicting the total reward at each time slice were just those features pertaining to the current time slice utterance. At the surface, this seems

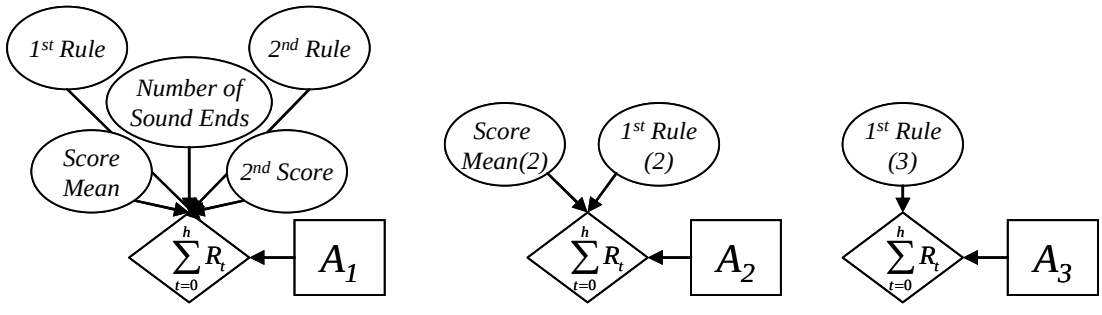

Figure 7. The learned cumulative total reward model where state space variables accumulate over time and they all predict the total reward for the entire dialogue.

to support the Markov assumption in that only the current state matters. However, this does not prove what is crucial about the assumption; namely, that future states need only depend on the current state and not previous states.

It is important to remember that the cumulative total reward model is model-free. It makes no assumptions about how features may be dependent on each other over time, but instead tries to predict the long term value of actions. In fact, while model-free approaches such as Q-learning (Watkins & Dayan, 1992) still learn policies using a system of equations similar to (4), a policy for the cumulative model is comparatively easy to learn. It is simply learning a decision tree that classifies what is likely to be the total reward for all the features it has observed. Even if the total reward were continuous, learning a regression is relatively easy. Furthermore, during runtime, it requires no inference, as do the other Markov models.

5 Implications and Limitations

Dialogue designers considering the use of reinforcement learning methods to learn policies for action selection in dialogue management could benefit from the implications of this experimental study. If learning an MDP is the goal, and the designer is uncertain about what variables to include in the state space, our results suggest that an MDP that includes just those variables that are parents of the local immediate reward performs just as well as a more complicated model with other variables that are not parents. In short, in factoring the state space, only those variables relevant to receiving a reward seem to matter.

On the other hand, if the designer is open to considering other types of models, the cumulative total reward model offers several advantages over the MDP. First, it is much simpler to learn from data and update. The cumulative model does not require estimation of the optimal value function nor inference. The policy is in the decision tree for the utility node. Moreover, if the reward function needs to be adjusted, it is much easier to update the total reward and re-learn the model than it is to re-compute the optimal policy. Second, the cumulative model may perform just as well if not better than the MDP. In our experiments, it indeed outperformed the MDP, though we are not quick to generalize this result due to the limitation of the study. And finally, the cumulative model makes fewer assumptions about the structure of the state space, which is quite appealing from a theoretical standpoint. It does not assume that that an optimal policy based just on the previous turn is as good as a policy based on the full history of interaction. Since the full accumulation of knowledge is expressed in the model, it accords well with sociolinguistic sensibilities.

The above implications should be moderated by the limitations of the study. First and foremost, the domain in which we learned the models was quite small with a relatively restricted command-and-control grammar, and a small action space. We plan to extend the simulation and learning methods to larger domains, and it will be interesting to see if our result still holds. Second, although we assiduously delineated every feature we could think of for the state space, we may not have included the "right" set of features that could have allowed the MDP or any other model to outperform the winner. Good feature engineering has been shown to make a significant difference in other natural-language processing domains, so it is prudent to remain humble about the features we utilized. Finally, our results depend on the techniques we used for learning the structure of the state space. We are agnostic about how these results may have turned out with other model selection techniques.

6 Conclusion

Spoken dialogue systems that learn optimal policies for dialogue management have typically utilized the MDP framework. In so doing, they are committed to the Markov assumption as a modeling choice, and very often, to assumptions about the structure of the state space. In this paper, we explored whether this choice is appropriate and empirically beneficial. In particular, we investigated whether constraining the state space by the Markov assumption truly affords the highest reward, especially when the structure of the state space may be unknown, which is typically the case. We examined four models that vary in terms of the extent of their temporal dependencies and found that the cumulative total reward model, a model-free approach that predicts the total reward for each time slice with state space features accumulating over time, outperformed the MDP and all other models in terms of total and average reward. This model had the smallest complexity by far, made the fewest number of assumptions, and is relatively easy to compute. For finite horizons, the cumulative total reward model offers an attractive alternative to the MDP.

References

Bellman, R.E. 1957. *Dynamic Programming*. Princeton University Press.

Chickering, D. 2002. "The WinMine Toolkit," *Microsoft Technical Report, MSR-TR-2002-103*.

Chickering, D., Heckerman, D., and Meeks, C. 1997. "A Bayesian Approach to Learning Bayesian Networks with Local Structure," *UAI-97*, pp. 80-89.

Chickering, D. and Paek, T. 2005. "Online Adaptation of Influence Diagrams," *Microsoft Technical Report, MSR-TR-2005-55*.

Clark, H. 1997. *Using Language*. Cambridge University Press. Cambridge University Press.

Heckerman, D. 1995. "A Bayesian Approach for Learning Causal Networks," *UAI-95*, pp. 285-295.

Howard, R., and Matheson, J. 1981. "Influence Diagrams," In Readings on the Principles and Applications of Decision Analysis, vol. 2. pp. 721-17. Strategic Decisions Group, Menlo Park, CA.

Kearns, M., Mansour, Y., Ng, A. 1999. "A Sparse Sampling Algorithm for Near-Optimal Planning in Large Markov Decision Processes," ICJAI-99.

Lauritzen, S.L. and Nilsson, D. 2001. "Representing and Solving Problems with Limited Information," Management Science, 47(9):1235-1251.

Levin, E., Pieraccini, R. and Eckert, W. 1998. "Using Markov Decision Processes for Learning Dialogue Strategies," *ICASSP-98*.

Meuleau, N., Hauskrecht, M., Kim, K., Peshkin, L., Kaelbling, L., Dean, T. and Boutilier, C. 1998. "Solving Very Large Weakly Coupled Markov Decision Processes," *AAAI-98*, pp. 165-172.

Paek, T. and Horvitz, E. 2004. "Optimizing Automated Call Routing by Integrating Spoken Dialog Models with Queuing Models," *HLT-NAACL-2004*, pp. 41-48.

Paek, T. and Horvitz, E. 2000. "Conversation as Action Under Uncertainty," *UAI-2000*, pp. 445-464.

Roy, N., Pineau, J. and Thrun, S. 2000. "Spoken Dialogue Management Using Probabilistic Reasoning," *ACL-2000*.

Singh, S., Litman, D., Kearns, M. and Walker, M. 2002. "Optimizing Dialogue Management with Reinforcement Learning: Experiments with the NJ-Fun System," *Journal of Artificial Intelligence Research*, 16: 105-133.

Sutton, R.S. and Barto, A.G. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

Tatman, J.A. and Shachter, R.D. 1990. "Dynamic Programming and Influence Diagrams," *IEEE Transactions on Systems, Man and Cybernetics*, 20(2):365-379.

Walker, M., Aberdeen, J., Boland, J., Bratt, E., Garofolo, J., Hirschman, L., Le, A., Lee, S., Narayanan, S., Papineni, K., Pellom, B., Polifroni, B., Potamianos, A., Prabhu, P., Rudnicky, A., Sanders, G., Seneff, S., Stallard, D., Wittaker, S. 2001(a). "DARPA Communicator Dialog Travel Planning Systems: The June 2000 Data Collection," *Eurospeech-2001*.

Walker, M.A., Passonneau, R.J. and Boland, J.E. 2001(b). "Quantitative and Qualitative Evaluation of DARPA Communicator Spoken Dialogue Systems," *ACL-2001*, pp. 515-522.

Watkins, C.J.C.H. and Dayan, P. 1992. "Q-Learning," *Machine Learning*, 8(3):229-256.

Young, S. 2000. "Probabilistic Methods in Spoken Dialogue Systems," *Philosophical Transactions of the Royal Society (Series A)*, 358(1769): 1389-1402.

Zhang, B., Cai, Q., Mao, J. and Guo, B. 2001. "Planning and Acting under Uncertainty: A New Model for Spoken Dialogue Systems," *UAI-2001*, pp. 572-579.