Using Wizard-of-Oz simulations to bootstrap Reinforcement-Learning-based dialog management systems

Jason D. Williams **Steve Young**
Department of Engineering, University of Cambridge, Cambridge, CB2 1PZ, United Kingdom
{jdw30,sjy}@eng.cam.ac.uk

Abstract

This paper describes a method for "bootstrapping" a Reinforcement Learning-based dialog manager using a Wizard-of-Oz trial. The state space and action set are discovered through the annotation, and an initial policy is generated using a Supervised Learning algorithm. The method is tested and shown to create an initial policy which performs significantly better and with less effort than a hand-crafted policy, which can be generated using a small number of dialogs.

1 Introduction and motivation

Recent work has successfully applied Reinforcement Learning (RL) to learning dialog strategy from experience, typically formulating the problem as a Markov Decision Process (MDP). (Walker et al., 1998; Singh et al., 2002; Levin et al., 2000). Despite successes, several open questions remain, especially the issue of how to create (or "bootstrap") the initial system prior to data becoming available from on-line operation.

This paper proceeds as follows. Section 2 outlines the core elements of an MDP and issues related to applying an MDP to dialog management. Sections 3 and 4 detail a method for addressing these issues, and the procedure used to test the method, respectively. Sections 5-7 present the results, a discussion, and conclusions, respectively.

2 Background

An MDP is composed of a state space, an action set, and a policy which maps each state to one action. Introducing a reward function allows us to create or refine the policy using RL. (Sutton and Barto, 1998).

When the MDP framework is applied to dialog management, the state space is usually constructed from vector components including information state, dialog history, recognition confidence, database status, etc. In most of the work to date both the state space and action set are hand selected, in part to ensure a limited state space, and to ensure training can proceed using a tractable number of dialogs. However, hand selection becomes impractical as system size increases, and automatic generation/selection of these elements is currently an open problem, closely related to the problem of exponential state space size.

3 A method for bootstrapping RL-based systems

Here we propose a method for "bootstrapping" a MDP-based system; specifically, we address the choice of the state representation and action set, and the creation of an initial policy.

3.1 Step 1: Conduct Wizard-of-Oz dialogs

The method commences with "talking wizard" interactions in which either the wizard's voice is disguised, or a Text-to-speech engine is used. We choose human/wizard rather than human/human dialogs as people behave differently toward (what they perceive to be) machines and other people as discussed in Jönsson and Dahlbick, 1988 and also validated in Moore and Browning, 1992. The dialog, including wizard's interaction with back-end data sources is recorded and transcribed.

3.2 Step 2: Exclude out-of-domain turns

The wizard will likely handle a broader set of requests than the system will ultimately be able to cover; thus some turns must be excluded. Step 2 begins by formulating a list of tasks which are to be included in the transcript; the remainder is labeled out-of-domain (OOD) and excluded.

This step takes an approach which is analogous to, but more simplistic than "Dialogue Distilling" (Larsson et al., 2000) which changes, adds and removes portions of turns or whole turns. Here rules simply stipulate whether to keep a whole turn.

3.3 Step 3: Enumerate action set and state space

Next, the in-domain turns are annotated with dialog acts. Based on these, an action set is enumerated, and a set of state parameters and their possible values to form a vector describing the state space is determined, including:

- Information state (e.g., departure-city, arrival-city) from the user and database.
- The confidence/confirmation status of information state variables.
- Expressed user goal and/or system goal.
- Low-level turn information (e.g., yes/no responses, backchannel, "thank you", etc.).
- Status of database interactions (e.g., when a form can be submitted or has been returned).

A variety of dialog-act tagging taxonomies exist in the literature. Here we avoid a tagging system that relies on a stack or other recursive structure (for example, a goal stack) as it is not immediately clear how to represent a recursive structure in a state space.

In practice, many information state components are much less important than their corresponding confirmation status, and can be removed.

Even with this reduction, the state space will be massive – probably too large to ever visit all states. We propose using a parameterized value function -- i.e., a value function that shares parameters across states (including states previously unobserved). One special case of this is state tying, in which a group of states share the same value function; an alternative is to use a Supervised Learning algorithm to estimate a value function.

3.4 Step 4: Form an initial policy

For each turn in the corpus, a vector is created representing the current dialog state plus the subsequent wizard action. Taking the action as the class variable, Supervised Learning (SL) is used to build a classifier which functions as the initial policy.

Depending on the type of SL algorithm used, it may be possible to produce a prioritized list of actions rather than a single classification; in this case, this list can form an initial list of actions permitted in a given state.

As noted by Levin et al. (2000), supervised learning is not appropriate for optimizing dialog strategy because of the temporal/environmental nature of dialog. Here we do not assert that the SL-learned policy will be optimal – simply that it can be easily created, and that it will be significantly better than random guessing, and better and cheaper to produce than creating a cursory handcrafted strategy.

3.5 Limitations of the method

This method has several obvious limitations:

- Because a talking, perfect-hearing wizard is used, no/little account is taken of the recognition errors to be expected with automated speech recognition (ASR).
- Excluding too much in Step 2 may exclude actions or state parameters which would have produced a superior deployed system.

4 Experimental design

The proposed approach has been tested using the Autoroute corpus of 166 dialogs, in which a talking wizard answered questions about driving directions in the UK (Moore and Browning, 1992).

A small set of in-domain tasks was enumerated (e.g., gathering route details, outputting summary information about a route, disambiguation of place names, etc.), and turns which did not deal with these tasks were labeled OOD and excluded. The latter included gathering the caller's name and location ("UserID"), the most common OOD type.

The corpus was annotated using an XML schema to provide the following:

- 15 information components were created (e.g., `from`, `to`, `time`, `car-type`).
- Each information component was given a status: `C` (Confirmed), `U` (Unconfirmed), and `NULL` (Not known).
- Up to 5 routes may be under discussion at once – the state tracked the route under dis-

cussion (RUD), total number of routes (TR), and all information and status components for each route.

- A component called flow tracked single-turn dialog flow information from the caller (e.g., yes, no, thank-you, silence).
- A component called goal tracked the (most recent) goal expressed by the user (e.g., plan-route, how-far). Goal is empty unless explicitly set by the caller, and only one goal is tracked at a time. No attempt is made to indicate if/when a goal has been satisfied.

33 action types were identified. Some of which take information slots as parameters (e.g., wh-question, implicit-confirmation).

The corpus gave no indication of database interactions other than what can be inferred from the dialog transcripts. One common wizard action asked the caller to "please wait" when the wizard was waiting for a database response. To account for this, we provided an additional state component which indicated whether the database was working called db-request, which was set to true whenever the action taken was please-wait and false otherwise. Other less common database interactions occurred when town names were ambiguous or not found, and no attempt was made to incorporate this information into the state representation.

The state space was constructed using only the status of the information slots (not the values); of the 15, 4 were occasionally expressed (e.g., day of the week) but not used to complete the transaction and were therefore excluded from the state space. Two turns of wizard action history were also incorporated. This formulation of the state space leads to approximately 10^{33} distinct states.

For evaluation of the method, a hand-crafted policy of 30 rules mapping state to action was created by inspecting the dialogs.[1]

5 Results

Table 1 shows in-domain vs. out-of-domain wizard and caller turns. Figures 1 through 4 show counts of flow values, goal values, action values, and state components, respectively. The most common action type was "please-wait" (14.6% of actions).

Turn type	Total	In domain	OOD: User ID	OOD: Other
Wizard	3155 (100%)	2410 (76.4%)	594 (18.8%)	151 (4.8%)
Caller	2466 (100%)	1713 (69.5%)	561 (22.7%)	192 (7.8%)

Table 1: In-domain and Out-of-domain (OOD) turns

Criteria	States	Visits
Visited only once	1182 (85.7%)	1182 (45.9%)
Visited more than once without a conflict	96 (7.0%)	353 (13.7%)
Visited more than once with conflict	101 (7.3%)	1041 (40.3%)
TOTAL	1379 (100%)	2576 (100%)

Table 2: "Conflicts" by state and visits

Estimated action probabilities	Visits
p(action taken | state) > p(any other action | state)	774 (74.3%)
p(action taken | state) = p(one or more other actions | state) > p(all remaining actions | state)	119 (11.4%)
p(action taken | state) < p(another action | state)	148 (14.2%)
TOTAL	1041 (100%)

Table 3: Estimated probabilities in "conflict" states

Engine	Class	Precision
jBNC	Action-type only	72.7%
	Action-type & parameters	66.7%
C4.5	Action-type only	79.1%
	Action-type & parameters	72.9%
Hand-craft	Action-type only	58.4%
	Action-type & parameters	53.9%

Table 4: Results from SL training and evaluation

In some cases, the wizard took different actions in the same state; we labeled this situation a "conflict." Table 2 shows the number of distinct states that were encountered and, for states visited more than once, whether conflicting actions were selected. Of states with conflicts, Table 3 shows probabilities estimated from the corpus.

The interaction data was then submitted to 2 SL pattern classifiers – c4.5 using decision-trees

[1] It was not clear in what situations some of the actions should be used, so some (rare) actions were not covered by the rules.

(Quinlan, 1992) and jBNC using Naïve Bayesians (Sacha, 2003). Table 4 shows 10-fold cross validation classification error rates classifying (1) the action type, and (2) the action type with parameters, as well as the results for the hand-crafted policy.

Figure 5 show the 10-fold cross validation classification error rates for varying amounts of training data for two different pattern classifiers and action-type/action-type and parameters.

6 Discussion

The majority of the data collected was "usable": although 26.7% of turns were excluded, 20.5% of these were due to a well-defined task not under study here (user identification), and only 6.1% fell outside of designated tasks. That said, it may be desirable to impose a minimum threshold on how many times a flow, goal, or action must be observed before adding it to the state space or action set given the "long tails" of these elements.

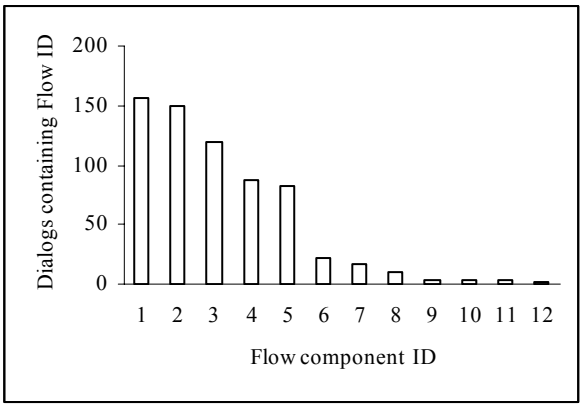

Figure 1: Dialogs containing flow components

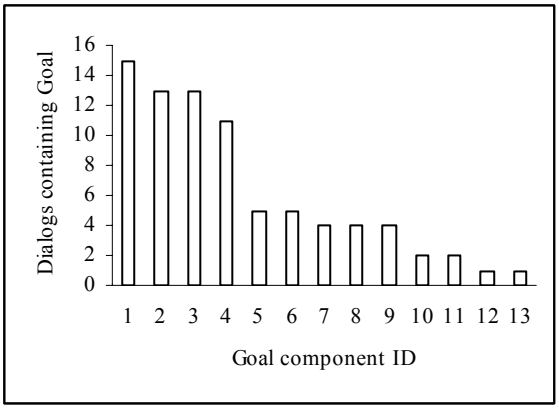

Figure 2: Dialogs containing goal components

About half of the turns took place in states which were visited only once. This confirms that massive amounts of data would be needed to observe all states which are within dialogs, and suggests dialog does not primarily visit familiar states.

Within a given state, the wizard's behavior is stochastic, occasionally deviating from an otherwise static policy. Some of this behavior results from database information not included in the corpus and state space; in other cases, the wizard is occasionally making random choices with no apparent basis.

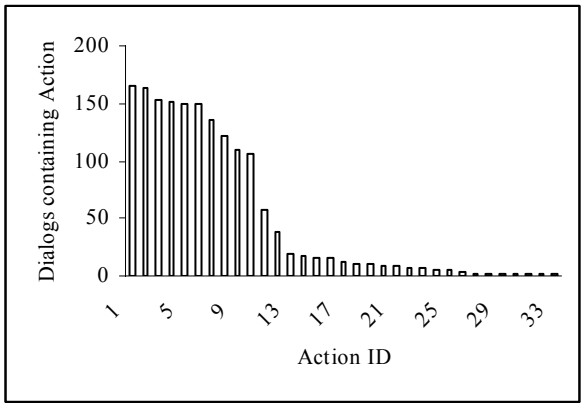

Figure 3: Dialogs containing action types

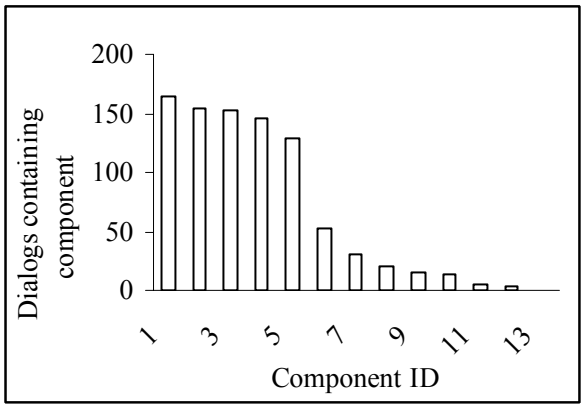

Figure 4: Dialogs containing information components

Figure 5 implies that a relatively small number of dialogs (several hundred turns, or about 30-40 dialogs) contain the vast majority of information relevant to SL algorithms – less than expected. Correctly predicting the wizard's action in 72.9% of turns is significantly better than the 58.4% correct prediction rate from the handcrafted policy.

When a caller allows the system to retain initiative, the policy learned by the c4.5 algorithm handled enquiries about single trips perfectly. Policy errors start to occur as the user takes more initiative, entering less well observed states.

Hand examination of a small number of misclassified actions indicate that about half of the actions were "reasonable" – e.g., including an extra item in a confirmation. Hand examination also confirmed that the wizard's non-deterministic behavior and lack of database information resulted in misclassifications.

Other sources of mis-classifications derived primarily from under-account of the user's goal and other deficiencies in the expressiveness of the state space.

7 Conclusion & future work

This work has proposed a method for determining many of the basic elements of a RL-based spoken dialog system with minimal input from dialog designers using a "talking wizard." The viability of the model has been tested with an existing corpus and shown to perform significantly better than a hand-crafted policy and with less effort to create.

Future research will explore refining this approach vis-à-vis user goal, applying this method to actual RL-based systems and finding suitable methods for parameterized value functions

References

A. Jönsson and N. Dahlbick. 1988. *Talking to A Computer is Not Like Talking To Your Best Friend.* Proceedings of the Scandinavian Conference on Artificial Intelligence '88, pp. 53-68.

Staffan Larsson, Arne Jönsson and Lena Santamarta. 2000. *Using the process of distilling dialogues to understand dialogue systems*. ICSLP 2000, Beijing.

Ester Levin, Roberto Pieraccini and Wieland Eckert. 2000. *A Stochastic Model of Human-Machine Interaction for Learning Dialogue Structures*. IEEE Trans on Speech and Audio Processing 8(1):11-23.

R. K. Moore and S. R. Browning. 1992. *Results of an exercise to collect 'genuine' spoken enquiries using Wizard of Oz techniques.* Proc. of the Inst. of Acoustics.

Ross Quinlan. 1992. *C4.5 Release 8.* (Software package). http://www.cse.unsw.edu.au/~quinlan/

Jarek P. Sacha. 2003. *jBNC version 1.0.* (Software package). http://sourceforge.net/projects/jbnc/.

Satinder Singh, Diane Litman, Michael Kearns, Marilyn Walker. 2002. *Optimizing Dialogue Management with Reinforcement Learning: Experiments with the NJFun System.* Journal of Artificial Intelligence Research, vol 16, 105-133.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning: an Introduction.* The MIT Press, Cambridge, Massachusetts, USA.

Marilyn A. Walker, Jeanne C. Fromer, Shrikanth Narayanan. 1998. *Learning Optimal Dialogue Strategies: A Case Study of a Spoken Dialogue Agent for Email.* Proc. 36th Annual Meeting of the ACM and 17th Int'l Conf. on Comp. Linguistics, 1345--1352.

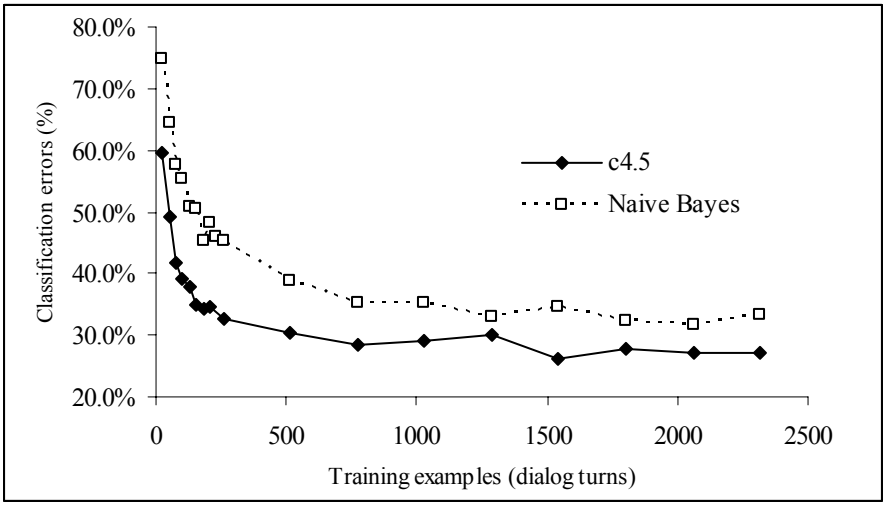

Figure 5: Classification errors vs. training samples for action-type & parameters