Agenda-Based User Simulation for Bootstrapping a POMDP Dialogue System

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye and Steve Young
Cambridge University Engineering Department
Trumpington Street, Cambridge, CB21PZ, United Kingdom
{js532, brmt2, kw278, hy216, sjy}@eng.cam.ac.uk

核心价值点在于构建了一个数据集的生成的方法

Abstract

This paper investigates the problem of bootstrapping a statistical dialogue manager without access to training data and proposes a new probabilistic agenda-based method for simulating user behaviour. In experiments with a statistical POMDP dialogue system, the simulator was realistic enough to successfully test the prototype system and train a dialogue policy. An extensive study with human subjects showed that the learned policy was highly competitive, with task completion rates above 90%.

1 Background and Introduction

1.1 Bootstrapping Statistical Dialogue Managers

One of the key advantages of a statistical approach to Dialogue Manager (DM) design is the ability to formalise design criteria as objective reward functions and to learn an optimal dialogue policy from real dialogue data. In cases where a system is designed from scratch, however, it is often the case that no suitable in-domain data is available for training the DM. Collecting dialogue data without a working prototype is problematic, leaving the developer with a classic chicken-and-egg problem.

Wizard-of-Oz (WoZ) experiments can be carried out to record dialogues, but they are often time-consuming and the recorded data may show characteristics of human-human conversation rather than typical human-computer dialogue. Alternatively, human-computer dialogues can be recorded with a handcrafted DM prototype but neither of these two methods enables the system designer to test the implementation of the statistical DM and the learning algorithm. Moreover, the size of the recorded corpus (typically $\ll 10^3$ dialogues) usually falls short of the requirements for training a statistical DM (typically $\gg 10^4$ dialogues).

1.2 User Simulation-Based Training

In recent years, a number of research groups have investigated the use of a two-stage simulation-based setup. A statistical user model is first trained on a limited amount of dialogue data and the model is then used to simulate dialogues with the interactively learning DM (see Schatzmann et al. (2006) for a literature review).

The simulation-based approach assumes the presence of a small corpus of suitably annotated in-domain dialogues or out-of-domain dialogues with a matching dialogue format (Lemon et al., 2006). In cases when no such data is available, handcrafted values can be assigned to the model parameters given that the model is sufficiently simple (Levin et al., 2000; Pietquin and Dutoit, 2005) but the performance of dialogue policies learned this way has not been evaluated using real users.

1.3 Paper Outline

This paper presents a new probabilistic method for simulating user behaviour based on a compact representation of the *user goal* and a stack-like *user agenda*. The model provides an elegant way of encoding the relevant dialogue history from a user's point of view and has a very small parameter set so that manually chosen priors can be used to bootstrap the DM training and testing process.

In experiments presented in this paper, the agenda-based simulator was used to train a statistical POMDP-based (Young, 2006; Young et al., 2007) DM. Even without any training of its model parameters, the agenda-based simulator was able to produce dialogue behaviour realistic enough to train a competitive dialogue policy. An extensive study[1] with 40 human subjects showed that task completion with the learned policy was above 90% despite a mix of native and non-native speakers.

[1]This research was partly funded by the EU FP6 TALK Project. The system evaluation was conducted in collaboration with O. Lemon, K. Georgila and J. Henderson at Edinburgh University and their work is gratefully acknowledged.

2 Agenda-Based Simulation

2.1 User Simulation at a Semantic Level

Human-machine dialogue can be formalised on a semantic level as a sequence of state transitions and dialogue acts[2]. At any time t, the user is in a state S, takes action a_u, transitions into the intermediate state S', receives machine action a_m, and transitions into the next state S'' where the cycle restarts.

$$S \rightarrow a_u \rightarrow S' \rightarrow a_m \rightarrow S'' \rightarrow \cdots \quad (1)$$

Assuming a Markovian state representation, user behaviour can be decomposed into three models: $P(a_u|S)$ for action selection, $P(S'|a_u, S)$ for the state transition into S', and $P(S''|a_m, S')$ for the transition into S''.

2.2 Goal- and Agenda-Based State Representation

Inspired by agenda-based methods to dialogue management (Wei and Rudnicky, 1999) the approach described here factors the user state into an agenda A and a goal G.

$$S = (A, G) \qquad \text{and} \qquad G = (C, R) \quad (2)$$

During the course of the dialogue, the goal G ensures that the user behaves in a consistent, goal-directed manner. G consists of constraints C which specify the required venue, eg. a centrally located bar serving beer, and requests R which specify the desired pieces of information, eg. the name, address and phone number (cf. Fig. 1).

The user agenda A is a stack-like structure containing the pending user dialogue acts that are needed to elicit the information specified in the goal. At the start of the dialogue a new goal is randomly generated using the system database and the agenda is initially populated by converting all goal constraints into *inform* acts and all goal requests into *request* acts. A *bye* act is added at the bottom of the agenda to close the dialogue.

As the dialogue progresses the agenda and goal are dynamically updated and acts are selected from the top of the agenda to form user acts a_u. In response to incoming machine acts a_m, new user acts are pushed onto the agenda and no longer relevant ones are removed. The agenda thus serves as a convenient way of tracking the progress of the dialogue as well as encoding the relevant dialogue history. Acts can also be temporarily stored when actions of higher priority need to be issued first, hence providing the simulator with a simple model of user memory[3].

[2]In this paper, the terms *dialogue act* and *dialogue action* are used interchangeably. The notation *act(a=x, b=y,...)* is used to represent a dialogue act of a given type *act* (such as *inform* or *request* with items $a = x$, $b = y$, etc.

[3]See Fig. 1, turns 1-3, for an example: System turn 1 *"What pricerange?"* triggers the user act *inform(pricerange=cheap)* to be pushed onto the agenda but it is not executed until turn 3 because *negate(drinks=beer)* is issued first.

$$C_0 = \begin{bmatrix} type = bar \\ drinks = beer \\ area = central \end{bmatrix}$$

$$R_0 = \begin{bmatrix} name = \\ addr = \\ phone = \end{bmatrix}$$

Sys 0 Hello, how may I help you?

$$A_1 = \begin{bmatrix} inform(type = bar) \\ inform(drinks = beer) \\ inform(area = central) \\ request(name) \\ request(addr) \\ request(phone) \\ bye() \end{bmatrix}$$

Usr 1 I'm looking for a nice bar serving beer.

Sys 1 Ok, a wine bar. What pricerange?

$$A_2 = \begin{bmatrix} negate(drinks = beer) \\ inform(pricerange = cheap) \\ inform(area = central) \\ request(name) \\ request(addr) \\ request(phone) \\ bye() \end{bmatrix}$$

Usr 2 No, beer please!

Sys 2 A bar serving beer, correct?

$$A_3 = \begin{bmatrix} affirm() \\ inform(pricerange = cheap) \\ inform(area = central) \\ request(name) \\ request(addr) \\ request(phone) \\ bye() \end{bmatrix}$$

Usr 3 Yeah something cheap in the town centre.

Sys 3 Murphy's on Main Square serves cheap beer.

$$A_4 = \begin{bmatrix} request(phone) \\ bye() \end{bmatrix}$$

Usr 4 Ok, and what's the phone number?

Sys 4 The number is 796 69 94.

$$A_5 = \begin{bmatrix} bye() \end{bmatrix}$$

Usr 5 Thanks, goodbye!

Figure 1: Sample dialogue and agenda sequence

2.3 User Act Selection

At any time during the dialogue, the updated agenda of length N contains all dialogue acts the user intends to convey to the system. Since the agenda is ordered according to priority, with $A[N]$ denoting the top and $A[1]$ denoting the bottom item, selecting the next user act simplifies to popping n items off the top of the stack. Hence, letting $a_u[i]$ denote the ith item in the user act a_u

$$a_u[i] := A[N - n + i] \quad \forall i \in [1..n], 1 \leq n \leq N. \quad (3)$$

Using $A[N–n+1..N]$ as a Matlab-like shorthand nota-

tion for the top n items on A, the action selection model becomes a Dirac delta function

$$P(a_u|S) = P(a_u|A, G) = \delta(a_u, A[N- n+1..N]) \quad (4)$$

where the random variable n corresponds to the level of initiative taken by the simulated user. In a statistical model the probability distribution over integer values for n should be conditioned on A and learned from dialogue data. For the purposes of bootstrapping the system, n can be assumed independent of A and any distribution $P(n)$ that places the majority of its probability mass on small values of n can be used.

2.4 State Transition Model

The factorisation of S into A and G can now be applied to the state transition models $P(S'|a_u, S)$ and $P(S''|a_m, S')$. Letting A' denote the agenda after selecting a_u (as explained in the previous subsection) and using $N' = N - n$ to denote the size of A', we have

$$A'[i] \quad := \quad A[i] \quad \forall i \in [1..N']. \quad (5)$$

Using this definition of A' and assuming that the goal remains constant when the user executes a_u, the first state transition depending on a_u simplifies to

$$\begin{aligned} P(S'|a_u, S) &= P(A', G'|a_u, A, G) \\ &= \delta(A', A[1..N'])\delta(G', G). \end{aligned} \quad (6)$$

Using $S = (A, G)$, the chain rule of probability, and reasonable conditional independence assumptions, the second state transition based on a_m can be decomposed into *goal update* and *agenda update* modules:

$$\begin{aligned} &P(S''|a_m, S') \\ &= \underbrace{P(A''|a_m, A', G'')}_{\text{agenda update}} \underbrace{P(G''|a_m, G')}_{\text{goal update}}. \end{aligned} \quad (7)$$

When no restrictions are placed on A'' and G'', the space of possible state transitions is vast. The model parameter set is too large to be handcrafted and even substantial amounts of training data would be insufficient to obtain reliable estimates. It can however be assumed that A'' is derived from A' and that G'' is derived from G' and that in each case the transition entails only a limited number of well-defined atomic operations.

2.5 Agenda Update Model for System Acts

The agenda transition from A' to A'' can be viewed as a sequence of push-operations in which dialogue acts are added to the top of the agenda. In a second "clean-up" step, duplicate dialogue acts, *null()* acts, and unnecessary *request()* acts for already filled goal request slots must be removed but this is a deterministic procedure so that it can be excluded in the following derivation for simplicity. Considering only the push-operations, the items 1 to N' at the bottom of the agenda remain fixed and the update model can be rewritten as follows:

$$\begin{aligned} &P(A''|a_m, A', G'') \\ &= P(A''[1..N'']|a_m, A'[1..N'], G'') \quad (8) \\ &= P(A''[N'+1..N'']|a_m, G'') \\ &\quad \cdot \ \delta(A''[1..N'], A'[1..N']). \quad (9) \end{aligned}$$

The first term on the RHS of Eq. 9 can now be further simplified by assuming that every dialogue act item in a_m triggers one push-operation. This assumption can be made without loss of generality, because it is possible to push a *null()* act (which is later removed) or to push an act with more than one item. The advantage of this assumption is that the known number M of items in a_m now determines the number of push-operations. Hence $N'' = N' + M$ and

$$\begin{aligned} &P(A''[N'+1..N'']|a_m, G'') \\ &= P(A''[N'+1..N'+M]|a_m[1..M], G'') \quad (10) \\ &= \prod_{i=1}^{M} P(A''[N'+i]|a_m[i], G'') \quad (11) \end{aligned}$$

The expression in Eq. 11 shows that each item $a_m[i]$ in the system act triggers one push operation, and that this operation is conditioned on the goal. This model is now simple enough to be handcrafted using heuristics. For example, the model parameters can be set so that when the item *x=y* in $a_m[i]$ violates the constraints in G'', one of the following is pushed onto A'': *negate(), inform(x=z), deny(x=y, x=z)*, etc.

2.6 Goal Update Model for System Acts

The goal update model $P(G''|a_m, G')$ describes how the user constraints C' and requests R' change with a given machine action a_m. Assuming that R'' is conditionally independent of C' given C'' it can be shown that

$$\begin{aligned} &P(G''|a_m, G') \\ &= P(R''|a_m, R', C'')P(C''|a_m, R', C'). \end{aligned} \quad (12)$$

To restrict the space of transitions from R' to R'' it can be assumed that the request slots are independent and each slot (eg. *addr, phone, etc.*) is either filled using information in a_m or left unchanged. Using $R[k]$ to denote the k'th request slot, we approximate that the value of $R''[k]$ only depends on its value at the previous time step, the value provided by a_m, and $\mathcal{M}(a_m, C'')$ which indicates a match or mismatch between the information given in a_m and the goal constraints.

$$\begin{aligned} &P(R''|a_m, R', C'') \\ &= \textstyle\prod_k P(R''[k]|a_m, R'[k], \mathcal{M}(a_m, C'')). \end{aligned} \quad (13)$$

To simplify $P(C''|a_m, R', C')$ we assume that C'' is derived from C' by either adding a new constraint, setting an existing constraint slot to a different value (eg. *drinks=dontcare*), or by simply changing nothing. The choice of transition does not need to be conditioned on the full space of possible a_m, R' and C'. Instead it can be conditioned on simple boolean flags such as "Does a_m ask for a slot in the constraint set?", "Does a_m signal that no item in the database matches the given constraints?", etc. The model parameter set is then sufficiently small for handcrafted values to be assigned to the probabilities.

3 Evaluation

3.1 Training the HIS Dialogue Manager

The Hidden Information State (HIS) model is the first trainable and scalable implementation of a statistical spoken dialog system based on the Partially-Observable Markov Decision Process (POMDP) model of dialogue (Young, 2006; Young et al., 2007; Williams and Young, 2007). POMDPs extend the standard Markov-Decision-Process model by maintaining a *belief space*, i.e. a probability distribution over dialogue states, and hence provide an explicit model of the uncertainty present in human-machine communication.

The HIS model uses a grid-based discretisation of the continuous state space and online ϵ-greedy policy iteration. Fig. 2 shows a typical training run over 60,000 simulated dialogues, starting with a random policy. User goals are randomly generated and an (arbitrary) reward function assigning 20 points for successful completion and -1 for every dialogue turn is used. As can be seen, dialogue performance (defined as the average reward over 1000 dialogues) converges after roughly 25,000 iterations and asymptotes to a value of approx. 14 points.

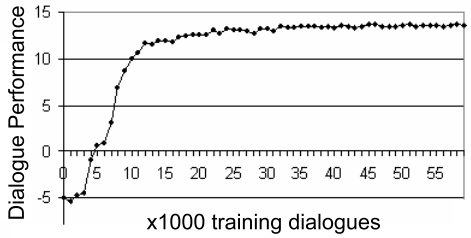

Figure 2: Training a POMDP system

3.2 Experimental Evaluation and Results

A prototype HIS dialogue system with a learned policy was built for the Tourist Information Domain and extensively evaluated with 40 human subjects including native and non-native speakers. A total of 160 dialogues with 21667 words was recorded and the average Word-Error-Rate was 29.8%. Task scenarios involved finding a specific bar, hotel or restaurant in a fictitious town (eg. the address of a cheap, Chinese restaurant in the west).

The performance of the system was measured based on the recommendation of a correct venue, i.e. a venue matching all constraints specified in the given task (all tasks were designed to have a solution). Based on this definition, 145 out of 160 dialogues (90.6%) were completed successfully, and the average number of turns to completion was 5.59 (if no correct venue was offered the full number of turns was counted).

4 Summary and Future Work

This paper has investigated a new agenda-based user simulation technique for bootstrapping a statistical dialogue manager without access to training data. Evaluation results show that, even with manually set model parameters, the simulator produces dialogue behaviour realistic enough for training and testing a prototype system. While the results demonstrate that the learned policy works well for real users, it is not necessarily optimal. The next step is hence to use the recorded data to train the simulator, and to then retrain the DM policy.

References

O. Lemon, K. Georgila, and J. Henderson. 2006. Evaluating Effectiveness and Portability of Reinforcement Learned Dialogue Strategies with real users: the TALK TownInfo Evaluation. In *Proc. of IEEE/ACL SLT*, Palm Beach, Aruba.

E. Levin, R. Pieraccini, and W. Eckert. 2000. A Stochastic Model of Human-Machine Interaction for Learning Dialog Strategies. *IEEE Trans. on Speech and Audio Processing*, 8(1):11–23.

O. Pietquin and T. Dutoit. 2005. A probabilistic framework for dialog simulation and optimal strategy learning. *IEEE Trans. on Speech and Audio Processing, Special Issue on Data Mining of Speech, Audio and Dialog*.

J. Schatzmann, K. Weilhammer, M.N. Stuttle, and S. Young. 2006. A Survey of Statistical User Simulation Techniques for Reinforcement-Learning of Dialogue Management Strategies. *Knowledge Engineering Review*, 21(2):97–126.

X Wei and AI Rudnicky. 1999. An agenda-based dialog management architecture for spoken language systems. In *Proc. of IEEE ASRU*. Seattle, WA.

J. Williams and S. Young. 2007. Partially Observable Markov Decision Processes for Spoken Dialog Systems. *Computer Speech and Language*, 21(2):231–422.

S. Young, J. Schatzmann, K. Weilhammer, and H. Ye. 2007. The Hidden Information State Approach to Dialog Management. In *Proc. of ICASSP*, Honolulu, Hawaii.

S. Young. 2006. Using POMDPs for Dialog Management. In *Proc. of IEEE/ACL SLT*, Palm Beach, Aruba.