A New Focus Strategy for Efficient Dialog Management

Xinqi Bao[1] Yunfang Wu[1(✉)] Xueqiang Lv[2]

[1] Key Laboratory of Computational Linguistics, Peking University, Beijing, 100871, China
[1] yikusitian1990@163.com; wuyf@pku.edu.cn
[2] Beijing Key Laboratory of Internet Culture and Digital Dissemination Research, Beijing
[2] lxq@bistu.edu.cn

Abstract. The dialog manager is the most important component for a dialog system, in which the dialog state tracking is crucial to a real-world system. We claim that the intractability of dialog states comes from two aspects: the large slot size in user's goal and the large candidate value size for each slot. For the first time, we propose a new focus strategy to deal with the former problem, by reducing the full slots of the user's goal into a small subset focus slot. We also implement a partition-based method to deal with the latter problem. Then we combine both strategies to take advantage of their complement property. In our experiment of a real-world application in an image purchase domain, our proposed focus strategy is far faster than both the partition method and the naïve algorithm with comparable quality.

1 Introduction

Dialog systems interact with human users via natural language to help them achieve some goals. The dialog manager is the most important component for a dialog system. Typically, it should deal with the state tracking task and system action generating task during the conversation. In the state tracking task, the dialog manager interprets what the user has said and updates some representations of the current dialog state, which encodes various dialog information including user's goal, current user's action and dialog history. Then in the action generating task, the dialog manager generates proper response action back to the user based on the current dialog state representation, which is then updated as in the first task.

In a dialog manger, the dialog state tracking is crucial because the system relies on it to know where the dialog goes and how to generate proper response. Conventional dialog managers use hand-crafted deterministic rules to interpret each dialog act and update the state. However, it is not easy to maintain a precise and accurate representation of the dialog state because there are always ambiguities and uncertainties in what the user said. What's more, natural language understanding techniques are not perfect, and thus it may cause the system misunderstanding the true goal of the user and taking a lot of efforts in error recovering.

Statistical approaches for dialog state tracking provide an opportunity for solving the above problems in a flexible way (Young, 2002). Early attempts model the dialog process as a Markov Decision Process (MDP) (Levin et al., 2000), which uses reinforcement learning for policy optimizing and provides a well-formed statistical framework allowing forward planning. However, MDP assumes fully observable dialog states and cannot deal with the uncertainty of those states. Partially Observable MDP (POMDP) (Williams et al., 2006) assumes that the state is not directly observable that reflects the uncertainty in the interpretation of user utterances. It maintains a distribution called belief state b(s) over all possible states rather than only the most likely one, and thus it explicitly models the uncertainty of many possible dialog states, which is more robust to recognizing errors and ambiguities.

However, it is not straightforward to construct a POMDP-based dialog manager for a real-world task because maintaining a full state space distribution will be intractable. Two types of views are proposed to achieve tractable and scalable implementation of POMDP-based state tracking. The first one tries to reduce the state space complexity by factoring the dialog state into independent components. For example, in a "slot filling" task, the whole dialog state space can be factored into subspaces of independent slots whose values are required to be filled. It becomes feasible to maintain a distribution over each individual slot (Williams and Young, 2007a, b). The second view is to retain a full belief state representation but only updating those most likely ones (Bohus et al., 2006; Williams, 2010). The intuition behind the second approach is that, during a conversation, most of the possible states are improbable or undistinguishable with each other, because they are not involved in the current dialog yet. So the dialog manager may safely neglect them and consider only the high probable ones to approximate the true state distribution. These two views do not conflict with each other. Actually, optimizing techniques from the second view takes advantages of careful independence assumptions related to the first view.

However, it is still not a trivial task to keep track of a full state space. For example, in a typical "slot filling" task with N slots to fill and K different candidate values for each slot, there are K^N possible value combinations. The state space size grows exponentially as N and K become large, so efficient state representation and manipulation are still crucial for a real-world application. We observe that the intractability of dialog states comes from two aspects: large factor size from state space decomposition (related to N) and large candidate value size of each factor (related to K).

The latter problem is partially resolved by clustering the undistinguishable states into partitions like Hidden Information State (HIS) model (Young et al., 2010). However, though many previous POMDP-based systems model the dialog process as a slot-filling task and factor the goal of a dialog into many slots, no work has paid attention to the first problem.

In this paper, we propose a method to solve the large factor size problem, namely dialog focus strategy. The intuition is that, not all factors of slots should be computed at a specific timestamp, and so neglecting some of them will avoid computing those beliefs of complex factors every time and thus reduce the time complexity greatly. Our strategy is more efficient and easy to implement than previous approaches. To the best of our knowledge, this is the first work to address the large factor size problem to build a tractable and scalable dialog manager. Also, we implement partition-based strategy (Thomson and Young, 2010; Young et al., 2010) into our framework to deal with the inner factor complexity. Finally, we conduct an ensemble method that combines both strategies to take advantage of their complement property. We fit our system into a real-world image purchase task, and evaluate different strategies using user simulations. Our focus strategy is far faster than the naive method and also beats the partition based strategy with comparable quality.

2 Related Work

In recent years in the field of dialog model, statistical dialog state tracking systems turn out to be more robust and scalable than hand-crafted ones (Bohus et al., 2006; Young et al., 2010; Thomson and Young, 2010). These methods try to maintain a distribution over different dialog state hypothesis to help the dialog manager choose the proper dialog strategy.

Bohus and Rudnicky (2006) train a conditional model in a discriminative fashion to estimate the distribution over a set of state hypotheses. They employ a large set of informative features to achieve high accuracy. However, they only keep track of a handful of state hypotheses thus the correct one may be discarded. Angeliki et al. (2013) exploits the structure of the dialog state hypothesis, and draws a rich set of features and further keeps a number of hypotheses-invariant features to allow an unlimited number of hypotheses.

Williams et al. (2006) propose a POMDP model that effectively represents the uncertainty of dialog states. Later, POMDP-based models are widely used as generative approaches in different domains, such as restaurant recommendations (Jurčíček et al., 2012), sightseeing recommendations (Misu et al., 2010), appointment scheduling (Georgila et al., 2010), etc. Some experiments apply it to the more difficult problem of learning negotiation policies (Heeman, 2009; Georgila and Traum, 2011a, b), and tutoring domains (Tetreault and Litman, 2008; Chi et al., 2011).

Many efforts have been made to solve the state space exploding problem in the POMDP framework. Thomson and Young (2010) presents a method based on the loopy belief propagation algorithm to make the state distribution updating tractable. They factor the state space into different components and use the marginal of these components as features for policy optimization. A grouped form of loopy belief propagation is implemented on factor graph and provides a significant reduction in calculation time. However, it is aimed at graph structure that contains cycles rather than the structure with many parallel components like in our task.

The HIS model (Young et al., 2010) groups similar user goals into equivalence classes called partitions, based on the assumption that all of the goals in the same partition are equally probable. The partitions are refined as the dialog progresses, and they are tree-structured to take account of the dependencies defined in the domain ontology. State tracking then requires only maintaining and updating relatively fewer partitions, which can be done in a real-world system. However, they do not consider the cases when there are many slots such that the number of partitions will grow exponentially with the slot size as the dialog progresses.

3 POMDP based dialog model

3.1 Basic mathematics

We first outline the mathematics of POMDP and explain our basic POMDP-based dialog model. A partially observable Markov decision process is formally defined as $(S;A;T;R;O;G;b_0)$, where S is a set of states; A is a set of actions; T defines a transition probability $P(s_t|s_{t-1},a_{t-1})$; R defines the expected reward $r(s_t,a_t)$; O is a set of observations; G defines an observation generating probability $P(o_t|s_t,a_{t-1})$ and b_0 is an initial belief state.

The POMDP operates as follows. At each timestamp t, the world is in an unobserved state s_t. Since s_t is not known exactly, a distribution over possible states called a belief state b_t is maintained, where $b_t(s_t)$ indicates the probability of being in a particular state s_t. The machine selects an action a_t by using its dialog policy receiving a reward r_t, and transfers to the unobserved next state s_{t+1}, where s_{t+1} depends only on s_t and a_t. The machine then receives an observation o_{t+1}, which is dependent on s_{t+1} and a_t.

3.2 Dialog Model

We now fit the POMDP into our dialog process. As in Thomson and Young (2010), the dialog state at each timestamp t is decomposed into three distinct types of information parts: the user's goal g_t, the act of the user utterance u_t and the dialog history h_t. The belief state b_t is then a joint distribution over these three components.

The user's goal encompasses the information that must be gleaned from the user in order to fulfil the task. It is further factored into several independent slots which reflect different aspects of the goal. For example, in a simple weather information system, the user's goal can be represented by 2 different slots: the city and the date. Thus for a sentence "I want to know the weather in Beijing today", the goal can be represented as {city:Beijing; date:today}. We denote the user's goal as g, with N_{goal} slots. Each slot is denoted as g_i and has N_i possible values v_{ik} $(k=0,1.. N_i)$.

The user's act u_t represents the type of dialog action. The act types we used are listed in Table 1, including 5 types of user's act and 5 types of server's act. In this paper, our goal is to investigate the belief updating strategies, so we define a simple set of act types. In the experiment, we use a SVM classifier to predict the user's act with a precision 96%.

The dialog history h_t tracks some information relating to previous turns. We model the history h_t in two aspects: *act history* and *slot history*. The *act history* encodes the previous acts made by the user and the system, such as the most recent server's act and the frequency of each act type. The *slot history* indicates the progress on each slot, which is represented by a four-state machine, as shown in Figure 1.

The dialog goes as follows (shown in Figure 2). At each timestamp t, the user chooses a dialog act u_t according to $P(u_t|g,u_{t-1},h_{t-1})$, and then he generates natural language output o_t according to the observation model $P(o_t|g,u_t)$. Next, the system chooses proper response to the user's act by some dialog policy, and updates the dialog states and goes into the next timestamp.

Table 1. The act types for user/system

Type	Description	Actor
Inform	provide information	User/ system
Yes	confirm the system's utterance	User
No	deny the system's utterance	User
Don't know	do not know about information	User
Chat	domain irrelevant chatting	User
Ask	ask the value of some slot	System
Confirm	confirm the value of some slots	System
List	list out candidate values of some slots	System
Pardon	ask for repetition	system

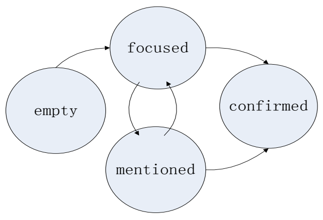

Fig. 2. The slot history in a four-state machine

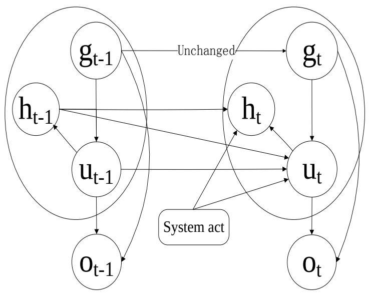

Fig. 1. The dialog process

The belief state for dialog history *h* is deterministically updated by heuristic rules. And the joint belief state $b_t(u_t,g)$ for the goal and act is calculated by Equation (1):

$$b_t(u_t, g) \sim P(o_t \mid u_t, g) \cdot P(u_t \mid g, u_{t-1}, h_{t-1}) \cdot b_{t-1}(g) \tag{1}$$

Our model is a little different from previous works mainly in the following three aspects:

1) Our system is designed for online dialog tasks where users type in the content by keyboard, and thus our user' inputs are textual sentences. So our model does not include Automatically Speech Recognizing (ASR) module and does not care sematic errors on ASR output.
2) The observation o_t is the user's natural language input instead of recognizing results from front-end recognizing modules. Thus, $P(o|g,u)$ directly models the process generating utterance from unobserved states, and there is no error propagation from frontend NLP modules.
3) In a real application, it is always assumed that the user does not change his goal in a dialog task. Most of previous work makes the same assumption. In our model,

we do not denote the goal's transition for simplicity, and the user's goal is now regarded as a global hidden state to be inferred.

3.3 Policy optimization

We select Natural Actor Critic (NAC) (Peters and Schaal, 2008) to learn the optimal dialog policy, which is also adopted by (Thomson and Young, 2010; Jurčíček et al., 2012; Teruhisa Misu et al., 2012).

At timestamp *t*, the system action set is sampled from a soft-max policy:

$$\pi(sa_t = k \mid \Phi, W) = \frac{e^{\sum_{i=1}^{I} \phi_i \cdot w_{ki}}}{\sum_{j=1}^{K} e^{\sum_{i=1}^{I} \phi_i \cdot w_{ji}}} \tag{2}$$

Here, $\Phi = (\phi_1, \phi_2, \phi_3 \ldots \phi_I)$ denotes the vector of feature functions representing the current dialog state and $W = (w_{11}, w_{12}, w_{13} \ldots w_{1I}, \ldots w_{JI})$denotes the policy parameters where w_{ji}measures the contribution of *i-th* feature to selecting the *j-th* action. The NAC optimizes over policy parameter *W* by solving linear regression problems.

The above training procedure requires thousands of dialog tracks to gather gradient information. So a simulated user (SU) (Georgila et al., 2006) that will behave similarly to a real user is built in order to interact with the policy to explore the search space and thus facilitate learning. We will describe our user simulator in the succeeding section.

4 Strategies for belief updating

It becomes intractable to directly implement the proposed POMDP-based dialog process when there are many slots and many candidate values for each slot. Although the state space is decomposed into subcomponents and the user's goal is further decomposed into independent slots, we still have to maintain $\sum_{i=1}^{N_{goal}} N_i$ marginal probabilities *P(goal*$_i$*=k)* of slot values. A straightforward naïve method is to exponentially enumerate over many possible slot value combinations as shown in Figure 3.

Thomson and Young (2010) use the Loopy Belief (LP) propagation to improve efficiency by exploiting the dependencies between graph components. But LP does not fit into our model because there can be many slot factors creating high costs in message passing.

In the next subsections, we will introduce our belief updating strategies that alleviate computation burden in maintaining marginal probabilities of slot values and current user's act. We observe that the intractability of dialog states comes from two aspects: the large slot size and the large candidate value size for each slot. Accordingly we implement two different methods to deal with the two problems, namely partition strategy and focus strategy. Further, an ensemble method combining both strategies is proposed to achieve better performance.

```
//initialization
For each user goal slot $g_i$:
    For each candidate value $v_{ij}$ of $g_i$:
        $b\_goal[i][j]_t = 0$
For each user's act type $u$:
        $b\_a[u]_t = 0$

//updating
For each user goal slot value combination:
    $g = \{g_1 = v_1, g_2 = v_2, ..., g_{N_{goal}} = v_{N_{goal}}\}$
    For each user's act type $u$:
        $P(u,g) \leftarrow P(o \mid u,g) \cdot P(u \mid g, u_{t-1}, h) \cdot P_{t-1}(g)$
        For each slot $g_i$:
            $b\_goal[i][v_i]_t \leftarrow b\_goal[i][v_i]_t + P(u,g)$
            $b\_a[u]_t \leftarrow b\_a[u]_t + P(u,g)$

//normalizing
Normalize the beliefs $b\_goal_t$ and $b\_a_t$
```

Fig. 3. The naïve algorithm

4.1 Partition Strategy

To deal with the large candidate value size problem, we follow the methodology of HIS model (Young et al., 2010). At any timestamp, the candidate value space for each slot of the user's goal can be divided into a number of equivalence classes called partitions, where the members of each class are equally possible and undistinguishable with each other.

Young et al. (2010) use a tree-based structure to represent the belief state and encode the policy in a nearest neighbor fashion. In order to utilize the feature-based log-linear policy sampling as in Equation (2), we need to maintain slot value marginal probability explicitly. Stronger dependency assumptions are made in our work.

Formally, we define values v_{i1}, v_{i2} of slot i belong to the same partition at some timestamp if and only if 1) they are of the same marginal belief 2) for any two slots value combinations g^1, g^2 :

$$g_i^1 = v_{i1},\ g_i^2 = v_{i2},\ g_{-i}^1 = g_{-i}^2 \tag{3}$$

and for any user's act type u:

$$\begin{aligned} P(o \mid u, g^1) &= P(o \mid u, g^2) \\ P(u \mid g^1, h) &= P(u \mid g^2, h) \end{aligned} \tag{4}$$

This means that the current collected information does not help distinguish between two values and so it is unnecessary to treat them separately.

Initially, all candidate values of a slot g_i are in a single root partition p_0. As the dialog progresses, this root partition is repeatedly split into smaller partitions.

If we can maintain the partitions of each slot dynamically, the belief updating process can be simplified by packaging the values of a partition and just enumerating over combinations of different partitions instead of the whole values. By assumption, we can randomly pick a value as the ‘representative’ of its belonged partition when calculating the joint belief *P(u,g)*, and then the marginal probability of a single value can be easily calculated by:

$$P(g_i = v) = \frac{P(partition_v)}{\# partition_v} \tag{5}$$

```
//split partitions
Split partitions based on the current user’s input.

//initialization
For each user goal slot g_i:
    For each partition_j of g_i:
        $b\_goal[i][j]_t = 0$
For each user’s act type u:
        $b\_a[t]_t = 0$

//updating
For each value partition combination
$p = \{g_1 \in partition_1, g_2 \in partition_2, ..., g_{N_{goal}} \in partition_{N_{goal}}\}$
    For each user’s act type u:
        Pick representative values from p
        $g = \{g_1 \in partition_1, ..., g_{N_{goal}} \in partition_{N_{goal}}\}$
        $P(u, p) \leftarrow P(o \mid u, g) \cdot P(u \mid g, h) \cdot P_{t-1}(g)$
    For each slot g_i:
        $b\_goal[i][partition_i]_t \leftarrow b\_goal[i][partition_i]_t + P(u, p)$
        $b\_a[u]_t \leftarrow b\_a[u]_t + P(u, p)$
//normalizing
Normalize the beliefs b_goal_t and b_a_t
Calculating marginal of single values
```

Fig. 4. The partition-based algorithm

Figure 4 illustrates the partition-based strategy. In practice, we associated each slot value with some keywords according to the domain specification documents. If a keyword is observed in user’s input then the associated values are spitted from its original partition to make up a new partition. For example, for the ‘usage’ slot in an image purchasing domain, the word ‘online’ triggers the system to separate those candidate usages from the original partition.

4.2 Focus Strategy

In many dialog systems, the number of candidate values of most of slots is quite small, and so the partition-based method does not fit them well, since it has to maintain partitions for all slots, which needs quite large extra costs. What's more, we still have to calculate the marginal probability for all values of all slots, which affects the computation efficiency taken by partitions, as illustrated in Figure 4.

In order to deal with the large slot size problem, we propose a focus-based strategy. The intuition is that when two people talk in a task-oriented conversation, they focus their attention only on a small portion that they know or are interested in. They concentrate on particular entities or particular perspectives on those entities (Grosz B J., 1978). So in our dialog model, we assume that the user tends to mention only a small proportion of slots in a turn. He may make a response to the server's query about certain slots or provide more information for some mentioned slots, but is not likely to refer all slots in a turn, because there is too much information when the slot size becomes large that he may not have enough knowledge to explicitly express them.

For example, a user wants to know the price of a service that has a lot of relevant parameters. It is unrealistic to expect that the user provides all the relevant information at once, because it is too complex for a non-expert to grasp all the knowledge. A good solution is to make a conversation between the server and the user. The server asks for these relevant parameters one by one through many dialog turns, and the user gives an explicit answer to each question.

Formally, a subset of slots $g_{\{k\}}=\{g_{k1}, g_{k2} \ldots, g_{km}\}$ is called the focus slots at a timestamp, if and only if for any two slot value combinations g^1, g^2:

$$g_i^1 = g_i^2 (i \in \{k_1, k_2 \ldots, k_m\}) \tag{6}$$

and for any user's act u:

$$\begin{aligned} P(o \mid u, g^1) &= P(o \mid u, g^2) = P(o \mid u, g_{\{k\}}) \\ P(u \mid g^1, h) &= P(o \mid g^2, h) = P(o \mid g_{\{k\}}, h) \end{aligned} \tag{7}$$

This means that those slots outside of the focus set contribute little to both the current user's dialog act and the natural language output.

If we can determine the focus slot set at each timestamp, the marginal $P(g_i=v)$ can be simplified as:

$$\begin{pmatrix} \sum_{u=1}^{N_u} \sum_{\{g_k\}_{-i}} P(o \mid u, \{g_{k_{-i}}, g_i = v\} \cdot P(u \mid \{g_{k_{-i}}, g_i = v\}, h) \cdot P_{t-1}(g_{k_{-i}}, g_i = v) & i \in \{k_1, k_2 \ldots k_m\} \\ P_{t-1}(g_i = v) & i \notin \{k_1, k_2 \ldots k_m\} \end{pmatrix} \tag{8}$$

We can see that the marginal probabilities of those slots outside of the focus set remain fixed, so there is no need to recalculate them in a turn. As a result, we only need calculating the marginal probabilities of those slots in the focus set, which only requires enumerating over values within the focus set.

The computation complexity now reduces from $O(|V|^n)$ to $O(|V|^m)$, where $|V|$ denotes the maximum size of each slot's candidate values, n is the full slot size, and m is the focus set's size which is generally observed to be less than 2 and is far smaller than n. Our focus strategy will work much faster than the above naïve method. What's more,

```
//get focus set
Generate the focus slots $\{g_{k_1}, g_{k_2}, ... g_{k_m}\}$ //initialization
Same as naïve method except that only the focus slots are estimated

//updating
For each focus slots value combination
   $g_{\{k\}} = \{g_{k_1} = v_1, g_{k_2} = v_2, ..., g_{k_m} = v_{k_m}\}$
   For each user's act type $u$:
   $P(u, g_{\{k\}}) \leftarrow P(o \mid u, g_{\{k\}}) \cdot P(u \mid g_{\{k\}}, h) \cdot P_{t-1}(g_{\{k\}})$
   For each slot $g_i$ in the focus set:
   $b\_goal[i][v_i]_t \leftarrow b\_goal[i][v_i]_t + P(u, g_{\{k\}})$ $b\_a[u]_t \leftarrow b\_a[u]_t + P(u, g_{\{k\}})$

//normalizing
Normalize the beliefs $b\_goal_t$ and $b\_a_t$
```

Fig. 5. The focus-based algorithm

```
//split partitions
Split partitions based on current user's input

//get focus set
Generate the focus slots $\{g_{k_1}, g_{k_2}, ... g_{k_m}\}$

//initialization
For each user goal slot $g_i$ in focus set
      For each partition$_j$ of $g_i$:
            $b\_goal[i][j]_t = 0$
      For each user's act type $u$:
            $b\_a[u]_t = 0$

//updating
For each partition value combination from focus slots
$p = \{g_1 \in partition_1, g_2 \in partition_2, ..., g_m \in partition_m\}$ For each user's act type $u$:
   Pick representative $g$ from $p$:
      $g = \{g_1 \in partition_1, ..., g_m \in partition_m\}$
      $P(u, p) \leftarrow P(o \mid u, g) \cdot P(u \mid g, h) \cdot P_{t-1}(g)$
    For each slot $g_i$:
$b\_goal[i][partition_i]_t \leftarrow b\_goal[i][partition_i]_t + P(u, p)$ $b\_a[u]_t \leftarrow b\_a[u]_t + P(u, p)$

//normalizing
Normalize the beliefs $b\_goal_t$ and $b\_a_t$
Calculating marginal of single value
```

Fig. 6. The combined method

it does not use dynamic structures to maintain all the values and thus expected to be also faster than partition strategy. Figure 5 demonstrates our revised focus-based algorithm.

In our experiments, we use some heuristics to determine the focus set at each timestamp. We build a domain keyword vocabulary from the domain documents, and each keyword is associated with one or more slots according to the domain description of these slots. For example, in an image-buying domain, the word ‘poster’ is associated with the ‘usage’ slot because it appears in the text describing the ‘usage’ of an image. To get the focus slot set, the dialog manager searches for these keywords in the user’s input sentence and marks the hit words with the associated slots. The slot referred by the last system’s act is also regarded as a focus slot, since the user tends to response to the system’s query and focus on the slot that the system has mentioned.

4.3 Combined Strategy

As noted before, the computation complexity comes from both the large slot size and large value size. In this section, we combine the two strategies to simultaneously reduce the complexity from both aspects. Figure 6 illustrates our combined strategy. The combined method takes advantage of the complement property of the two optimizing strategies. Factoring the user’s goal into independent slots makes it possible to maintain independent value partitions for each slot and only update partitions for focus slots.

5 Experiments

This section reports the experimental results of our proposed models and belief updating strategies. We first introduce the specific domain that our system applied to. Then we describe our user simulator that interacts with the system to train the dialogue policy.

5.1 The domain

Our system is designed for price inquiring service for Getty Images China, who is a leading supplier of images for business and consumers in China. It operates a commercial website which allows clients to search for images and purchase them. Costs of images vary greatly according to different factors which often confuse the purchasers, thus there needs an intellective server agent to automatically gather those factors and tell clients the final price. Of course it is friendlier to interact with purchasers/users with natural language dialog rather than providing a large, confusing and tedious HTML form to them.

The user's goal is represented by a table of different factors, which should be filled in during the conversation and requires the clients to provide relevant information in order to calculate the price. There are 6 slots in the user's goal table, each of which has a number of candidate values, as listed in Table 2.

Table 2. The slots and values

Slots	Candidate values
Usage	poster/advertisement/book/…
Time Span	1 year/2 year/5 year/…
Amount	10/100/500/1000/…
Size	1M/5M/15M/48M/…
Position	cover/inner page/wall…
Authority	electrical/non-electrical

5.2 User simulator

A user simulator simulates the user's behavior to train the dialog policy. At the start of a dialog, the system randomly generates a set of values for each slot which are treated as the real goal of the simulated user. The simulator then interacts with the system and provides relevant information based on the system's act. We constrain the dialog length to 20 turns, and if the system has not collected all necessary information within 20 turns the dialog should be forced to stop. The simulator then gives a score to the completed dialog according to certain metrics. The training process runs totally 80 iterations. In each iteration, it contains two steps: in simulating step 300 dialogs are simulated and then the optimization step updates policy parameters with the collected information of (dialog, score) pairs in the first step.

5.3 Computation Efficiency

We use 4 belief updating strategies stated above to train the model policy. The naïve method is the baseline, and the partition strategy mainly follows the idea of HIS and so also is regarded as a baseline. For each strategy, we simulate dialogs between the server agent and the user simulator, as mentioned in the above subsection.

We run our experiments on an Intel i5 Core machine with 4GB memory. The average simulating time per dialog of each of 4 methods are shown in Table 3. The naïve method is much slower than the other three strategies, with 1.7 seconds per dialog. The partition strategy takes 57.33 milliseconds (ms) per dialog. The focus strategy takes only 5.02 ms per dialog, which is 11 time faster than the partition strategy and 20 times faster than the naïve algorithm. The combined method is as fast as 2.38 ms per dialog, which is 24 time faster than the partition strategy. We can see that the focus strategy greatly reduces the computation time compared with the partition strategy, that is because as the dialog progresses, the size of partitions grows larger which makes the process slow down. As expected, combing the two strategies obtain the best efficiency.

The average reward trends of three optimized strategies are shown in Figure 7. It shows that after 40 iterations they all achieve high rewards (about 85) per dialog, so the convergence speed of three strategies is satisfying.

Table 3. The computation time of four strategies

Strategy	Time per dialog
Naïve	1.7s
Partition	57.33ms
Focus	5.02ms
Combined	2.38ms

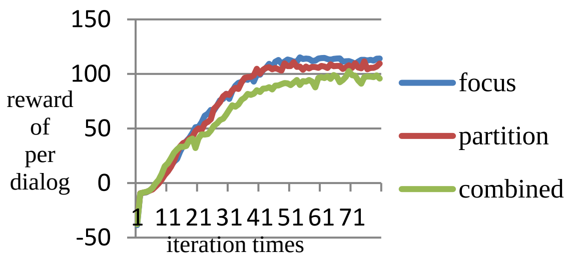

Fig. 7. The convergence curve of 3 strategies

6 Conclusion

In this paper, we aim to address the state space exploding problem for efficient dialog management in a POMDP framework. We claim that the intractability of dialog states comes from two aspects: the large slot size and the large candidate value size for each slot. In this paper, we propose a focus strategy to alleviate the computation complexity, by reducing the full slots into a small subset focus slot. We also implement the partition-based method similar to HIS model, with some adaptation in our task. We then combine both strategies to get better performance. We apply our systems to a real task of image purchasing, and conduct evaluation both with a user simulator and real users. Our focus strategy is far faster than partition-based method with comparable quality, and our combined method gets the best performance both in the computation time and the quality evaluated by human testers.

Acknowledgement
This work is supported by National Natural Science Foundation of China (61371129), National High Technology Research and Development Program of China (2015AA015403), Humanity and Social Science foundation of Ministry of Education (13YJA740060), and the Opening Project of Beijing Key Laboratory of Internet Culture and Digital Dissemination Research (ICDD201402).

References

1. Bohus D, Rudnicky A. 2006. A K-hypotheses+ Other Belief Updating Model. Proceedings of the AAAI Workshop on Statistical and Empirical Methods in Spoken Dialogue Systems.

2. Chi M, VanLehn K, Litman D, et al. 2011. An evaluation of pedagogical tutorial tactics for a natural language tutoring system: A reinforcement learning approach. *International Journal of Artificial Intelligence in Education*, 21(1): 83-113.
3. Georgila K, Henderson J, Lemon O. 2006. User simulation for spoken dialogue systems: learning and evaluation. *Interspeech*.
4. Georgila K, Traum D R. 2011. Reinforcement Learning of Argumentation Dialogue Policies in Negotiation. *Interspeech*.
5. Georgila K, Traum D. 2011. Learning culture-specific dialogue models from non culture-specific data. *Universal Access in Human-Computer Interaction. Users Diversity*. Springer Berlin Heidelberg.
6. Georgila K, Wolters M K, Moore J D. 2010. Learning dialogue strategies from older and younger simulated users. *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue. Association for Computational Linguistics*.
7. Grosz B J. 1978. Focusing in dialog. *Proceedings of the 1978 workshop on Theoretical issues in natural language processing. Association for Computational Linguistics*.
8. Heeman P A. 2009. Representing the reinforcement learning state in a negotiation dialogue. *Automatic Speech Recognition & Understanding*.
9. Jurčíček F, Thomson B, Young S. 2012. Reinforcement learning for parameter estimation in statistical spoken dialogue systems. *Computer Speech & Language*, 26(3): 168-192.
10. Levin E, Pieraccini R, Eckert W. 2000. A stochastic model of human-machine interaction for learning dialog strategies. *Speech and Audio Processing, IEEE Transactions on*, 8(1): 11-23.
11. Metallinou A, Bohus D, Williams J D. 2013. Discriminative state tracking for spoken dialog systems. *Proceedings of Annual Meeting of the Association for Computational Linguistics.*
12. Misu T, Georgila K, Leuski A, et al. 2012. Reinforcement learning of question-answering dialogue policies for virtual museum guides. *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue. Association for Computational Linguistics*.
13. Misu T, Kawahara T. 2010. Bayes risk-based dialogue management for document retrieval system with speech interface. *Speech Communication*, 52(1): 61-71.
14. Peters J, Schaal S. 2008. Natural actor-critic. *Neurocomputing*, 71(7): 1180-1190.
15. Schatzmann J, Weilhammer K, Stuttle M, et al. 2006. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The Knowledge Engineering Review*, 21(02): 97-126.
16. Tetreault J R, Litman D J. 2008. A reinforcement learning approach to evaluating state representations in spoken dialogue systems. *Speech Communication*, 50(8): 683-696
17. Thomson B, Young S. 2010. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech & Language*, 24(4): 562-588.
18. Williams J D, Young S. 2007a. Partially observable Markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2): 393-422.
19. Williams J D, Young S. 2007b. Scaling POMDPs for spoken dialog management. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(7): 2116-2129.
20. Williams J D. 2010. Incremental partition recombination for efficient tracking of multiple dialog states. *Acoustics Speech and Signal Processing (ICASSP), IEEE International Conference on*.
21. Williams J D. 2006. Partially observable Markov decision processes for spoken dialogue management. The doctoral thesis.

22. Young S, Gašić M, Keizer S, et al. 2010. The hidden information state model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech & Language*, 24(2): 150-174.
23. Young S. 2002. The statistical approach to the design of spoken dialogue systems. *Technical Report CUED/F-INFENG/TR.433, Cambridge University*.