Scalable summary-state pomdp hybrid dialog system for multiple goal drifting requests and massive slot entity instances

Sangjun Koo, Seonghan Ryu, Kyusong Lee, Gary Geunbae Lee

Abstract One of the main problems with Partially Observable Markov Decision Process (POMDP) in development of spoken dialog system (SDS) is lack of scalability. In development of an SDS with Electronic Program Guide (EPG) domain, we devised a POMDP approach which is operated with summary spaces to respond accurately to multiple drifting goals and massive numbers of slot entities. The main point of the proposed approach is to introduce a hybrid architecture that is implemented by a meta-action selector and a service provider. A trained POMDP policy was used to select meta-actions. The selected meta-actions were transformed to the system action in the service provider, which is implemented with the given system action model. By using this architecture, various system actions could be elicited with reduced complexity in the dialog process. We trained the system with the specified simulator and observed its behavior with learning curves in the Korean EPG domain. The convergence of learning curve implies the feasibility of our approach in commercial EPG domain SDS.

Key words: Partially Observable Markov Decision Process, meta-action selector, service provider

1 Introduction

A main challenge in development of Spoken Dialog Systems (SDSs) is to ensure a certain level of accuracy of system responses to ambiguous user input. The input can be affected by several factors including classification er-

Sangjun Koo, Seonghan Ryu, Kyusong Lee, Gary Geunbae Lee
Pohang University of Science and Technology, San 31, Pohang, Republic of Korea.
e-mail: {giantpanda,ryush,kyusonglee,gblee}@postech.ac.kr

rors committed by Automatic Speech Recognition (ASR) / Spoken Language Understanding (SLU).

Construction of an SDS that can respond to ambiguous input can be modeled as a decision-making problem under uncertainty. One of the most widely-used decision-making techniques is the Partially Observable Markov Decision Process (POMDP) framework which is derived from the Markov Decision Process (MDP) [1]. The POMDP framework is suitable for designing SDSs [2, 4, 6, 12, 14] because it allows SDS designers to elicit mathematical models of user behavior, and uses automated stochastic process to train SDSs [3].

However, several problems obstruct the use of the POMDP framework in SDSs. One of the major problems is lack of scability: the number of POMDP states increases exponentially as the number of named entitie increases; this trend results in complexity that is too high for a real-time SDS [3]. Two major trends of research have been conducted to solve this problem [12]: (1) Factoring slot information into sub-states to reduce the number of POMDP states [13], and (2) Constructing partitions of given dialog states to prune out irrelevant states [10, 11, 12].

Both approaches are based on the hypothesis that given initial goals of users do not easily change, and that if the goals change, their alterations can be described using simple rules. This assumption is valid in domains such as the Tourist Information Guide (TIG), in which the intention of a user can be represented as a single specific goal. However, when we designed an SDS for the Electronic Program Guide (EPG) domain, we observed that the hypothesis is not effective as they are in the TIG domain for three reasons: (1) Multiple drifting goals usually appear in a single dialog session, since most EPG dialogs consist of a series of independent requests and users do not have a specific goal in their mind; (2) The number of slot entity instances in the EPG domain is much larger than in the TIG domain. Entity slots in EPG typically cover open-category values including titles of TV programs and movies; (3) The dialog length is shorter in the EPG domain than in the TIG domain: most of the user requests are processed in a single request session.

In development of the SDS for EPG domain, we designed an architecture that is specified to operate with multiple drifting goals. The main idea of the proposed SDS is to separate overall dialog management (DM) into two components: meta-action selector and service provider. The meta-action selector was implemented with the POMDP framework to select meta-actions from given belief states; the service provider was implemented as a rule-based DM to select system actions for the user. The partition between two components reduces the number of dialog states, thereby reducing the complexity of the dialog process.

2 Overall architecture

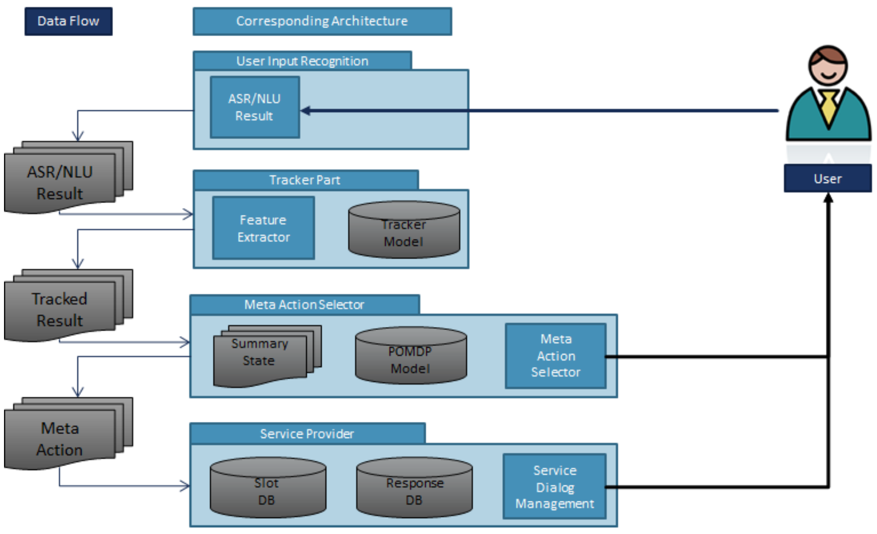

Fig. 1 Architecture of proposed SDS for EPG domain

The main architecture (Fig. 1) consists of three sub-components: Tracker, Meta-action selector and Service provider.

WUser input given by ASR/SLU is interpreted as a probability distribution of user intention slots. The dialog system considers this probability distribution to be an observation. *Tracker* tracks belief states of each entity slot individually and summarizes them into summary states. Summary states are compressed forms of original dialog states, and are introduced to guide selection of POMDP actions. *Meta-action selector* uses a trained POMDP policy model to select meta-actions with tracked summary states. These meta-actions include 'submit to service provider', 'confirm the value of slot x', 'request that the user provide the value of slot x'. If the 'submit' action is selected by the meta-action selector, *Service provider* elicits the corresponding system action to provide the appropriate service to the user.

3 Tracking of summary states

Let $o \in O$ be an observation of user input, let $b \in B$ be the internal state of given dialog system and let $a \in A$ be system output . The process of tracking summary states can be divided into two sub-tasks: (1) Updating internal belief state with the given previous state, the user input and system output. This subtask can be represented as a conditional probability form $P(b'|b,a,o)$. (2) Constructing summary states from original internal belief states. This subtask can be represented as a mapping function $\phi(b') : B \rightarrow \bar{B}$.

3.1 Update of goal states

A single tracker cannot easily track overall belief states. For example, 10 slots with 100 slot values each may yield as a total of 100^{10} states, which are unlikely to be tracked within reasonable time constraints. However, tracking each slot independently [8] is a reasonable strategy. Without loss of generality, suppose a given system includes slots x and y. Then the belief probability $b : b(s)$ can be represented as

$$b(s) = b(s_x)b(s_y). \tag{1}$$

By indexing each entity slot $i = 1, \ldots, N$, marginal slot states can be stated as $b_i : b(s_i)$. Belief states for user intention should also be tracked. We introduced the concept of slot UI with belief states b_{UI} to represent the value of the user intention slot. The joint belief probability $b(s_{UI}, s_i, s_j)$ can be represented as:

$$b(s_{UI}, s_i, s_j) = b(s_{UI})b(s_i|s_{UI})b(s_j|s_{UI}). \tag{2}$$

To acquire exact goal states, a full Bayesian network must be established to track the set of overall states. Although we consider a slot-independent model for update to ignore dependencies between slots, its computational complexity could increase exponentially with the number of possible observed slots and system output types. We used a heuristic method that uses update rules to track individual states. The main advantage of the heuristic method is that its complexity is reasonable and that the method does not need prior dialog examples for training. Let H be a heuristic function built using update rules [9]. Let the initial distribution of s be s_{init} and let the service action set be $A_{service}$. The update of belief probability $b \rightarrow b' : b(s'|s,a,o)$ can be represented as:

$$b(s'|s,a,o) = \begin{cases} H(s,a,o) & (a \notin A_{service}) \\ s_{init} & (a \in A_{service}). \end{cases} \tag{3}$$

3.2 Construction of summary states

As the number of overall states could be large, learning a feasible POMDP policy may be a difficult task. In the proposed system, we introduced summary states that would be feature states in training POMDP policy. The motivation of using summary states is to prune out redundant information to reduce the number of states.

We used grid-based approximation [8] to generate a summary state. The key idea of the grid approach is to use only information that is relevant to the two most-likely values, s_{x1} and s_{x2} for each slot x. The distribution between s_{x1} and s_{x2} can be approximated using grid points. Euclidean distance between grid points is measured and the nearest grid is selected for estimation. Appropriate selection of grid points is important. Previous work [8] selected seven grid points for the distribution $l(s_{x1}, s_{x2})$: [(1.0, 0.0), (0.8, 0.2), (0.8, 0.0), (0.6, 0.4), (0.6, 0.2), (0.6, 0.0), (0.4, $*$)]. We selected 22 grid points to increase the accuracy of the approximation. We also considered two extra conditional features in constructing summary state for each slots: Argument Relation feature and None-Value feature.

The Argument-Relation feature represents whether the given slot could be the appropriate argument with a given user intention. Suppose that there exists a slot "Actor name" of which values are the possible names of actors. "Actor name" can be an argument of the user intention "search-program", because users can utter names of actors to find a program. However, the slot cannot be an argument of the user intention, "volume-up", because users do not utter names when issuing this instruction.

The None-Value feature represents whether the most-likely value of given slot is "none", the special value which represents that the slot is not mentioned by the user. The motivation is to represent the case that instances of the slot are not likely to be uttered by a user. In this case, the system should elicit a meta-action to request the value of slot, rather than ask users whether the value of slot is "none". Let the number of grid points be N_l[1]. The size of the feature vector assigned to the None-Value feature is also N_l; this vector represents a bicomponent distribution l with the value $s_{x1} = none$.

The size of feature vector for each entity slot for one meta-action would be $N_c = 2N_l + 1$. Let $y_{i,l}$ be an indicator for corresponding grid points and let y_{none} and y_{arg} be indicators for the None-Value feature and the Argument-Relation feature, respectively. The kth value of $\phi(b_i)$ for the j-th meta action can be constructed as (Eq. 4). The construction is achieved by tiling corresponding features.

$$\phi_j(b_i)_k = \begin{cases} \begin{cases} y_{i,l} & \text{if } y_{arg} = 1 \text{ and } y_{none} = 0 \\ 0 & otherwise \end{cases} & \text{if } 0 \le k < N_l \\ \begin{cases} y_{i,l} & \text{if } y_{arg} = 1 \text{ and } y_{none} = 1 \\ 0 & otherwise \end{cases} & \text{if } N_l \le k < 2N_l \\ \begin{cases} 1 & \text{if } y_{arg} = 0 \\ 0 & otherwise \end{cases} & \text{if k} = 2N_l \end{cases} \tag{4}$$

[1] The notataion is suggested by the previous research[8]

4 Selection of meta-actions

The function of the meta-action selector is to choose which meta-action should be elicited for given tracked states. Meta-actions can be divided into two groups : *Submission* and *Confirmations*. When the 'Submission' meta-action is invoked, the action selector accepts the slot values from the tracker and submits them to the service-provider to generate a service system action. Otherwise, the action selector generates a corresponding meta-response and sends it to the user for confirmation. The 'Confirmations' include 'request user to repeat', 'request user to utter the value of specified slot x' and 'confirm user whether the value of slot x is y' (Table 1).

Table 1 Interpretation of meta-action

Meta Action (a_{meta})	Interpretation
$repeat$	requests the user to repeat
$request - x$	reqeusts the user to repeat the value of slot x
$confirm - x$	confirms the user whether the value of slot x has 1-best value
$submit$	accepts slot values from the tracker

To acquire adequate meta-actions, POMDP policies must be optimized with given rewards. The policies can be described as a parameter vector θ in Gibbs sampling (Eq. 5) and they can be calculated by any feasible reinforcement-learning technique. To acquired optimized POMDP policies, we used Episodic Natural Actor Critic algorithm [5, 8], which is a gradient descent method that uses gradients from the Fischer metric.

$$\pi(a|\bar{b},\theta) = \frac{e^{\theta \cdot \phi_a(b)}}{\sum_{a'} e^{\theta \cdot \phi_{a'}(b)}} \tag{5}$$

5 Generation of system actions

Meta-actions elicited by the optimized policy must be interpreted before they are provided to end-users, because meta-actions are internal messages for the system and do not include any information for users. Tracked result of belief states are accepted by the service-provider (Section 4). The service provider selects a system action by using dialog model, which can be denoted as $P(a_m|o,h)$ where h is dialog history. When the model selects an appropriate action to be performed, the service provider uses a template database to generate a response sentence. The service provider also uses accpeted dialog information to generate queries for content database to fill slot values (Fig. 2).

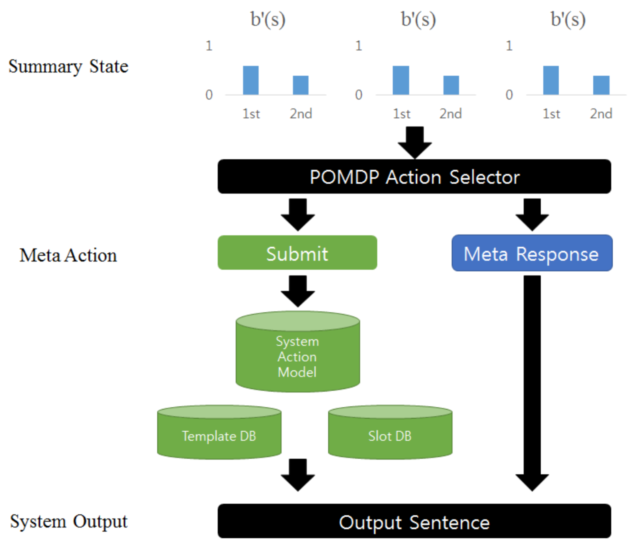

Fig. 2 Generation of system action using meta-action

This separated architecture allows the system to provide various service actions. Previous architectures required that SDS designers build new a tracker model and that they assign new reward values when new system actions were added to a system. In contrast, the proposed approach requires only modification of the dialog model in the service-provider. The main point of the suggested architecture is to bind valid system actions in an equivalent class in terms of performing service. This technique would be also effective for domains that have chracteristics that are similar to those of the EPG domain.

6 Experiment

6.1 Setting

To verify the feasibility of proposed architecture, we iteratively trained POMDP model and observed its learning curve to assure its convergence. We measured average reward and turn length for each batch in the training process. Each training sessions started with policy parameters initialized to a zero vector. Each dialog batch consisted of 100 dialog instances. Two hundred batches were provided to the system and policy parameters were updated at the end of each batch. Each dialog instance consists of two, three or four requests with corresponding constraints. At most 20 turns were permitted for each dialog instance.

The overall dialog was penalized for incorrect behaviors: requesting irrelevant slots, providing incorrect service and wrongly confirming. Penalties differed for each case, but ranged from -5 to -40. A reward of +15 was given if the system gave correct responses. Otherwise, the reward was not given. The initial reward was set to zero for each dialog instance.

For training and evaluating, we used a simulator that is implemented by a request-constraint model [7]. Because the overall dialog session consists of multiple independent requests with drifting goals, we reconstructed the model of this simulator to make it appropriate for use in a multiple-goal SDS. In the original design, each of requests shares its constraint in a global constraint set. In our design, they are supported by isolated constraint sets, dividing request independently. ASR/NLU error rates were 0.1 for both user intention slots and for entity slots.

6.2 Result

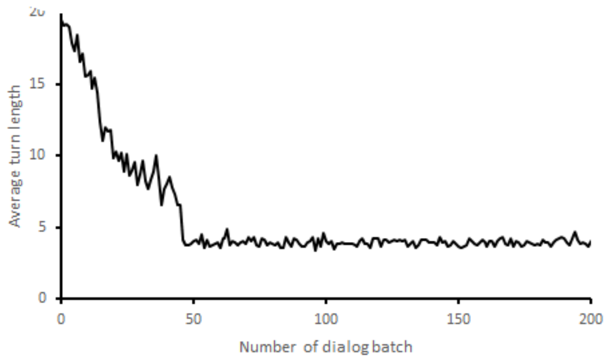

Fig. 3 Learning curves of average turn length

In the training sessions, the learning curves showed that the average reward and the average length of turn converged to certain level. Average length of turn converged to 4 (Fig. 3). Considering that the number of initial requests for each dialog instances ranged from 2 to 4, the converged level of average reward suggests that the proposed system can generate correct system actions for each user request with a managable number of confirmation sentences. Average reward for each dialog instance converged to 0, which is slightly less than optimal reward 30 - 60. Since penalties for inappropriate

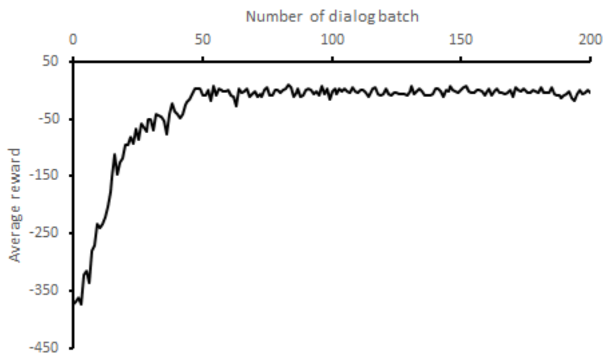

Fig. 4 Learning curves of average reward

confirmation (up to -40) were significantly larger than the reward value for a correct response, the converged reward lay in a predictable range. Reward is expected to increase if penalties are relieved.

Convergence of both learning curves indicate that the proposed architecture operates in stabilized manner with appropriate decision making process. It implies the feasibility of the overall suggested system, although further user-based experiment would be necessary to verify its utility in a direct manner.

7 Conclusion

This paper presents a new hybrid approach combined with the composite summary state POMDP components and stabilized service DM. The paper also introduces techniques to map summary states, which are used for POMDP action selection.

One further topic is a domain adaptation. Since the system was developed to overcome specific technical problems in the EPG domain, the proposed system is not guaranteed to operate effectively in other domains. Additional generalized components and models should be constructed in order to reflect characteristics of dialog acquired from other domains.

Another interesting topic would be to adopt iterative methods. By using a simulator and the proposed POMDP hybrid DM, log data of dialog could be generated. These log data could then be used to train the proposed DM and to construct an error model in simulator. We anticipate that iterative learning using real log data would boost the accuracy of overall dialog process. Further work will focus on methods to develop organized methods for these suggested iterative learning techniques.

Acknowledgements This work was supported by National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIP) (NRF-2014R1A2A1A01003041).

References

1. Bellman, R.: A markovian decision process. Indiana Univ. Math. J. **6**, 679–684 (1957)
2. Bui, T., Zwiers, J., Poel, M., Nijholt, A.: Toward affective dialogue modeling using partially observable markov decision processes. In: D. Reichardt, P. Levi, J.J. Meyer (eds.) Proceedings of the 1st workshop on Emotion and Computing ? Current Research and Future Impact, pp. 47–50. University of Bremen, Bremen (2006). URL http://doc.utwente.nl/66792/
3. Bui, T.h., Poel, M., Nijholt, A., Zwiers, J.: A tractable hybrid ddn-pomdp approach to affective dialogue modeling for probabilistic frame-based dialogue systems. Nat. Lang. Eng. **15**(2), 273–307 (2009)
4. Gasic, M., Breslin, C., Henderson, M., Kim, D., Szummer, M., Thomson, B., Tsiakoulis, P., Young, S.: Pomdp-based dialogue manager adaptation to extended domains. In: Proceedings of the SIGDIAL 2013 Conference, pp. 214–222. Association for Computational Linguistics, Metz, France (2013). URL http://www.aclweb.org/anthology/W/W13/W13-4035
5. Peters, J., Schaal, S.: Natural actor-critic. Neurocomputing **71**(79), 1180 – 1190 (2008). DOI http://dx.doi.org/10.1016/j.neucom.2007.11.026. URL http://www.sciencedirect.com/science/article/pii/S0925231208000532. Progress in Modeling, Theory, and Application of Computational Intelligenc 15th European Symposium on Artificial Neural Networks 2007 15th European Symposium on Artificial Neural Networks 2007
6. Roy, N., Pineau, J., Thrun, S.: Spoken dialogue management using probabilistic reasoning. In: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, ACL '00, pp. 93–100. Association for Computational Linguistics, Stroudsburg, PA, USA (2000). DOI 10.3115/1075218.1075231. URL http://dx.doi.org/10.3115/1075218.1075231
7. Schatzmann, J., Thomson, B., Weilhammer, K., Ye, H., Young, S.: Agenda-based user simulation for bootstrapping a pomdp dialogue system. In: Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers, NAACL-Short '07, pp. 149–152. Association for Computational Linguistics, Stroudsburg, PA, USA (2007)
8. Thomson, B., Young, S.: Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems. Comput. Speech Lang. **24**(4), 562–588 (2010). DOI 10.1016/j.csl.2009.07.003. URL http://dx.doi.org/10.1016/j.csl.2009.07.003
9. Wang, Z., Lemon, O.: A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In: Proceedings of the SIGDIAL 2013 Conference, pp. 423–432. Association for Computational Linguistics, Metz, France (2013). URL http://www.aclweb.org/anthology/W13-4067
10. Williams, J.D., Young, S.: Partially observable markov decision processes for spoken dialog systems. Comput. Speech Lang. **21**(2), 393–422 (2007)
11. Young, S.: Using pomdps for dialog management. In: Spoken Language Technology Workshop, 2006. IEEE, pp. 8–13 (2006). DOI 10.1109/SLT.2006.326785
12. Young, S., Gašić, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., Yu, K.: The hidden information state model: A practical framework for pomdp-based spoken dialogue management. Comput. Speech Lang. **24**(2), 150–174 (2010). DOI 10.1016/j.csl.2009.04.001. URL http://dx.doi.org/10.1016/j.csl.2009.04.001
13. Young, S., Schatzmann, J., Weilhammer, K., Ye, H.: The hidden information state approach to dialog management. In: Acoustics, Speech and Signal Processing, 2007.

ICASSP 2007. IEEE International Conference on, vol. 4, pp. IV–149–IV–152 (2007). DOI 10.1109/ICASSP.2007.367185
14. Zhang, B., Cai, Q., Mao, J., Guo, B.: Planning and acting under uncertainty: A new model for spoken dialogue systems. In: Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence, UAI'01, pp. 572–579. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001). URL http://dl.acm.org/citation.cfm?id=2074022.2074092