A Frame-Based Probabilistic Framework for Spoken Dialog Management Using Dialog Examples

Kyungduk Kim, Cheongjae Lee, Sangkeun Jung and Gary Geunbae Lee
Department of Computer Science and Engineering
Pohang University of Science & Technology (POSTECH)
San 31, Hyoja-Dong, Pohang, 790-784, Republic of Korea
{getta, lcj80, hugman, gblee}@postech.ac.kr

Abstract

This paper proposes a probabilistic framework for spoken dialog management using dialog examples. To overcome the complexity problems of the classic partially observable Markov decision processes (POMDPs) based dialog manager, we use a frame-based belief state representation that reduces the complexity of belief update. We also used dialog examples to maintain a reasonable number of system actions to reduce the complexity of the optimizing policy. We developed weather information and car navigation dialog system that employed a frame-based probabilistic framework. This framework enables people to develop a spoken dialog system using a probabilistic approach without complexity problem of POMDP.

1 Introduction

A robust dialog manager is an essential part of spoken dialog systems, because many such systems have failed in practice due to errors in speech recognition. Speech recognition errors can be propagated to spoken language understanding (SLU), so the speech input must be considered error-prone from a standpoint of dialog management. Therefore robust dialog managers are necessary to develop practical spoken dialog systems.

One approach to dialog management uses the partially observable Markov decision process (POMDP) as a statistical framework, because this approach can model the uncertainty inherent in human-machine dialog (Doshi and Roy, 2007). The dialog manager uses a probabilistic, rather than deterministic, approach to manage dialog. As more information becomes available, the dialog manager updates its belief states. A POMDP-based dialog manager can learn the optimized policy that maximizes expected rewards by reinforcement learning.

But applying classic POMDP to a practical dialog system incurs a scalability problem. The computational complexity of updating belief states and optimizing the policy increases rapidly with the size of the state space in a slot-filling dialog task. To solve this scalability problem, the method of compressing states or mapping the original state space to summarized space can be used (Williams and Young, 2006; Roy et al.,2005), but these algorithms tend to approximate the state space excessively. The complexity problem of POMDP comes from updating beliefs that are out of the user's intention, and from calculating the reward of system actions that do not satisfy user's objective.

In this paper, we propose a new probabilistic framework for spoken dialog management using dialog examples. We adopted a frame-based belief state representation to reduce the complexity of belief update. Furthermore, we used an example-based approach to generate only a reasonable number of system action hypotheses in a new framework. We developed a dialog system by using our new framework in weather information service and car navigation service.

Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue, pages 120–127,
Columbus, June 2008.

2 Overview

We try to address two problems of applying POMDP to slot-filling dialog management. 1) Computational complexity of belief update: it is difficult to maintain and update all belief states at every turn of dialog since there are too many dialog states in slot-filling dialog tasks. 2) Computational complexity of policy optimizing: optimizing complexity depends on both the space size of dialog states, and the number of available machine actions. In slot-filling dialog tasks, a system action can have various slot values so that the system needs to choose an action among a large number of action hypotheses.

In our new probabilistic framework (Figure 1), we try to solve these problems. Our approach uses 1) the frame-based belief state representation to solve the computational complexity problem of belief update and 2) the dialog examples to generate action hypotheses to solve the computational complexity of policy optimizing by reducing the number of system action hypotheses. First, the system groups belief states dynamically using frame-based belief state representation according to user's utterance and its SLU result. Then the system uses an example-based approach to generate only system action hypotheses that are suitable for current belief states. If there are too many hypotheses for calculating expected utility, the system prunes them away until only a reasonable number of hypotheses remains. The following describes the details of each system's component and the dialog managing process.

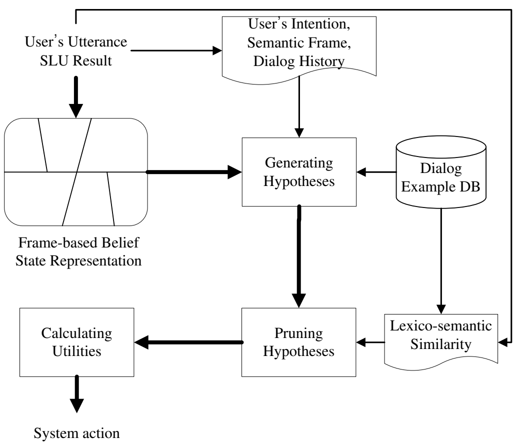

Figure 1. Overview of the system operation. Bold arrows indicate the control flow. Thin arrows indicate the data flow.

3 Frame-based Belief State Representation

We assumed that the machine's internal representation of the dialog state s_m consists of three components: user's goal s_u, user's last action a_u and dialog history s_d. This section briefly describes the basic introduction of POMDP framework and explains each component of machine's internal state in the standpoint of our frame-based probabilistic framework.

3.1 POMDP for spoken dialog management

A POMDP is defined as a tuple that consists of six substates: (S, A, P, R, Ω, O) where S is a set of state, A is a set of action, P is a transition probability $P(s'|s,a)$, R is a reward function $R(s,a,s')$, Ω is a set of observation and O is an observation model $P(o|s,a)$. The current state is not deterministic in a POMDP framework while it is determined as a specific state in a Markov decision process (MDP) framework. In a POMDP, the probability distribution over all states $s \in S$, which is referred as a belief state $b(s)$, is maintained instead of deterministic state. At each time instant t, the system chooses an action $a \in A$, and this causes the system to move from current state s to next state s' with the transition probability $P(s'|s,a)$. Then, the system is granted a reward $R(s,a)$ while the system receives an observation o with probability of $P(o|s',a)$. The system computes the belief state in the next time instance $b'(s')$ as a following:

$$b'(s') = k \cdot P(o \mid s,a) \sum_s P(s' \mid s,a) b(s)$$

where k is a normalizing factor. This process is referred as belief update.

Optimizing a POMDP policy is a process of finding a mapping function from belief states to actions that maximizes the expected reward. The system should compute a value function over belief spaces to find optimized actions. However, unlike as in a MDP, each value in a POMDP is a function of an entire probability distribution and belief spaces are very complex, so that a POMDP has a scale problem of computing the exact value function.

A POMDP for spoken dialog system is well formulated in (Williams and Young, 2007). First, a state s can be factored to three substates: (s_u, a_u, s_d)

where s_u is a user goal state, a_u is a user action, and s_d is a dialog history. A system action a_m and user action a_u can be cast as action a and observation o respectively. With some independence assumption between variables, the belief update equation can be rewritten as following:

$$
\begin{aligned}
b'(s') &= b(s'_u, a'_u, s'_d) \cdot \\
&= k \cdot P(\tilde{a}'_u \mid a_u) P(a_u \mid s'_u, a_m) \cdot \\
&\quad \sum_{s_u} P(s'_u \mid s_u, a_m) \cdot \sum_{s_d} P(s'_d \mid a'_u, s_d, a_m) \cdot \\
&\quad \sum_{a_u} b(s'_u, a'_u, s'_d),
\end{aligned}
$$

where $\tilde{a}'_u$ is an automatic speech recognizer (ASR) and SLU recognition result of user action. In our framework, belief update is done based on this equation. But applying this directly to a spoken dialog system can have a problem because the probabilities used in the equation are hard to estimate from the corpus due to the data sparseness. Therefore, we adopted Young's (2007) belief update formula that is simplified from the original equation.

3.2 User goal state

In a slot-filling dialog system, the user's goal can be represented as a fully-filled frame in which all slots of the frame contain values specified by the user's intention. Therefore, if a dialog system has W slots and each slot can have a value among V candidates, then V^W user goals can be represented as frames. This means that the number of user goals is related exponentially to the number of slots. This number of user goals is intractable in practical dialog systems.

Therefore, a method is needed to reduce the size of the state space rather than maintaining all belief states. To do this, we developed a frame-based belief state representation in which the system dynamically groups set of equivalent states to a high-level frame state. Frame state, which is a similar concept to the partition in the hidden information state (HIS) approach (Young et al, 2007) represents the indistinguishable classes of user's goals. The biggest difference between frame-based representation and partition-based representation is that the former uses only user input to split the frame state, whereas the latter uses the user input and external ontology rules such as a prior probability for belief of split partition. Therefore, the frame-based representation has relatively high domain portability because it does not need that kind of external domain dependent information.

In the frame-based belief state representation, a partially-filled frame state represents the current user's goal state for which the unfilled slot can be filled in the future, while a fully-filled frame state represents a complete user's goal state. Figure 2 describes an example of the subsumption relationship between partially filled frames and fully filled frames.

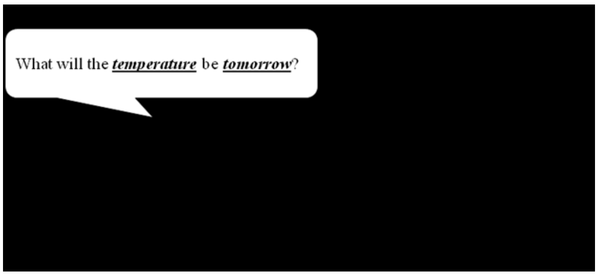

Figure 2. Subsumption relationship between partially filled frame and fully filled frame. The left frame is partially filled and three frames in the right side are fully filled.

At the start of a dialog, all states belong to the root frame state f_0. As the dialog progresses, this root frame state is split into smaller frame states whenever the value of a slot is filled by the user's input (Figure 3). First, if the user's input *[A=a]* fills the slot of the root frame state f_0, then it splits into two frame states: f_1, which includes all user goal states with the slot A having *'a'* as a value; and $\{f_0\text{-}f_1\}$, which is the relative complement of f_1. Next, if the user's input *[B=b]* is entered to the system, each frame f_1 and $\{f_0\text{-}f_1\}$ is split into smaller frame states. The system updates not all belief states but only the beliefs of the frame states, so that the computational complexity remains relatively small.

If each user's goal has uniform distribution, the belief of frame state $b(f)$ can be calculated as follows:

$$b(f) = \frac{\#\text{ of user goals contained in frame } f}{\#\text{ of all user goals}}$$

This can be computed as follows:

$$b(f) = \prod_{s_i \in S_{filled}} \frac{1}{|V_{s_i}|} \cdot \prod_{s_j \in S_{notFilled}} \frac{|V_{s_j} - V'_{s_j}|}{|V_{s_j}|},$$

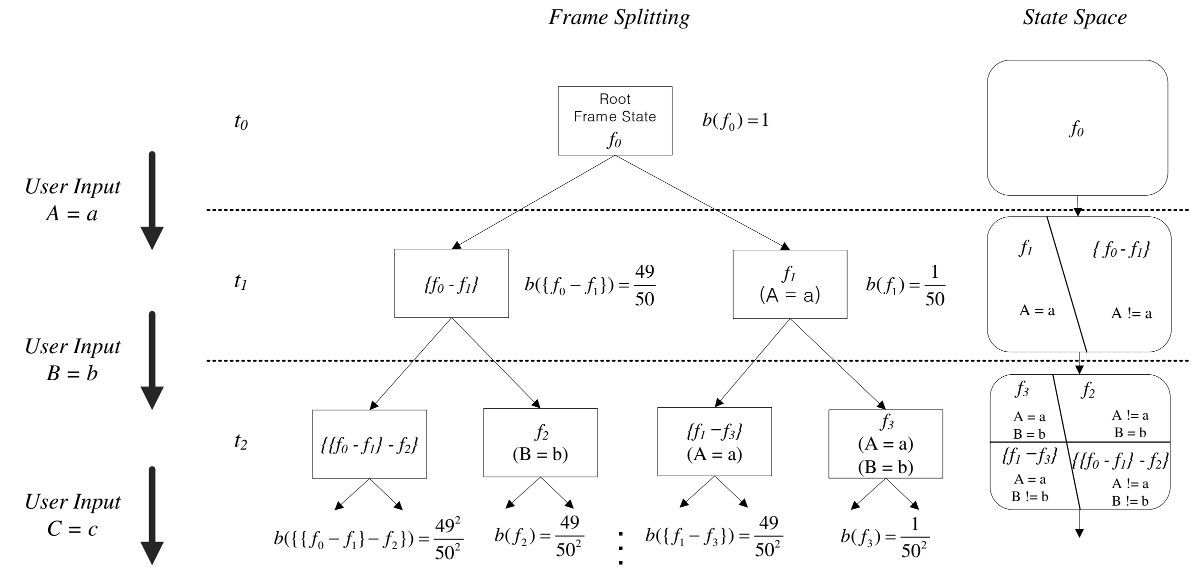

Figure 3. Splitting frame states and their beliefs with three user's inputs. f_0, f_1, f_2, … denote frame states and *b(f)* means the belief of frame state *f*. A, B, C are the slot labels and a, b, c are the respective values of these slots.

where S_{filled} means the set of slots that are filled by the user's input in frame state *f*, and $S_{notFilled}$ means the set of empty slots. V_s denotes the set of available values for slot *s*, and V_s' stands for the set of values for slot *s* that were specified by the user in other frame states.

3.3 User action

The SLU result of current user's utterance is used for the user action. The result frame of SLU consists of a speech act, a main goal, and several named-entity component slots for each user's utterance. The speech act stands for the surface-level speech act per single utterance and the main goal slot is assigned from one of the predefined classes which classify the main application actions in a specific domain such as "search the weather (SEARCH_WEATHER)" or "search the temperature (SEARCH_TEMPERATURE)" in the weather information service domain. The tasks for filling the named-entity component slots, such as, name of the city, name of the state, are viewed as a sequence labeling task. The Figure 4 shows some examples of predefined classes for SLU semantic frame in weather information service dialog system

Our SLU module was developed based on the concept spotting approach, which aims to extract only the essential information for predefined meaning representation slots, and was implemented by applying a conditional random field model (Lee et al., 2007).

Figure 4 Example predefined classes for semantic frame of SLU in weather information service dialog system.

3.4 Dialog history

Similar to the traditional frame-based dialog management approach, a frame can represent the history of the dialog. The difference between the traditional frame-based dialog manager and our framework is that traditional frame-based dialog manager maintains only one frame while our framework can maintain multiple dialog hypotheses. Moreover, each hypothesis in our framework can have a probability as in the belief state of the classic POMDP.

4 Example-based System Action Generation

4.1 Example-based system action hypothesis generation

It is impossible to consider all of the system actions as hypotheses because the number of possible actions is so large. We used an example-based approach to generate a reasonable number of system action hypotheses as hinted in (Lee et al., 2006). In this approach, the system retrieves the best dialog example from dialog example database (DEDB) which is semantically indexed from a dialog corpus. To query a semantically close example for the current situation, the system uses the user's intention (speech act and main goal), semantic frame (component slots) and discourse history as search key constraints (Lee et al., 2006). These search keys can be collected with SLU output (e.g., user intention and semantic frame) and discourse history in a dialog manager. Figure 5 describes an example of search key for DEDB on a weather information service system.

User's utterance	What will the ***temperature*** be ***tomorrow***? *Weather_Type Time_Date*
Search key constraints	Speech Act = wh_question Main Goal = search_temperature WEATHER_TYPE = 1 (filled) TIME_DATE = 1 (filled) LOC_CITY = 0 (unfilled) LOC_STATE = 0 (unfilled)
Lexico-semantic Input	What will the [WEATHER_TYPE] be [TIME_DATE]?

Figure 5. Example search key constraints for dialog example database.

For each frame state $f_1, \ldots, f_n$, the system generates one or more system action hypotheses by querying the DEDB respectively. Queried actions may inconsistent with the current frame state because the situation of indexed dialog examples may different from current dialog situation. Therefore, the system maps the contents of dialog example to information of current frame state. Slot values of frame state and information from content database (e.g., weather information database) are used for making the action consistent. If the system retrieves more than a threshold number of system action hypotheses using the search key constrains, then the system should prune away dialog examples to maintain only a reasonable number of hypotheses. We used lexico-semantic similarity between the user utterance and the retrieved examples to limit the number of hypotheses. To measure the lexico-semantic similarity, we first replace the slot values in the user utterance by its slot names to generate lexico-semantic input, and calculate the normalized edit distance between that input and retrieved examples (Figure 5). In the normalized edit distance, we defined following cost function $C(i,j)$ to give a weight to the term which is replaced by its slot name.

$$C(i,j) = \begin{cases} 0 \ \text{ if } w_{1,i} = w_{2,j} \\ 1 \ \text{ if } w_{1,i} \neq w_{2,j} \text{ and } w_{1,i}, w_{2,j} \notin S_{slot\_name} \\ 1.5 \ \text{ if } w_{1,i} \neq w_{2,j} \text{ and } w_{1,i}, w_{2,j} \in S_{slot\_name} \end{cases}$$

where $w_{1,i}$ is ith word of user's utterance, $w_{2,j}$ is jth word of dialog example's utterance, and S_{slot_name} is the set of slot names. According to the lexico-semantic similarity, the system appends the top N_h-ranked hypotheses to the final action hypotheses (where N_h is the rank threshold).

Many existing systems used heuristics or rule-based approaches to reduce the number of system action hypotheses (Young et al., 2007). But these methods are not flexible enough to handle all dialog flows because a system developer should design new heuristics or rules whenever the system needs to support a new kind of dialog flow. The example-based approach, on the contrary, can instantly refine the control of dialog flows by adding new dialog examples. This is a great advantage when a system developer wants to change or refine a dialog control flow.

4.2 Calculating Expected Utilities

We adopted the principle of maximum expected utility to determine the optimized system actions among the hypotheses (Paek and Horvitz, 2004).

$$\tilde{a}_m^* = \arg\max_a EU(a \mid \xi)$$
$$= \arg\max_a \sum_h P(H = h \mid \xi) u(a,h)$$
$$= \arg\max \sum_h b(h) u(a,h)$$

where ξ denotes all information about the environment, $u(a,h)$ means the utility of taking an action when the internal state of the machine is h, which consists of three substates, (f, a_u, s_d) : f is a frame state, a_u is a user's last action, and s_d is a dialog history. The utility function $u(a,h)$ can be specific to each application. We defined a handcrafted utility function to calculate the expected utility.

5 Experiments

We performed two evaluations. 1) Real user evaluation: we measured the user satisfaction with various factors by human. 2) Simulated user evaluation: we implemented user simulator to measure the system performance with a large number of dialogs. We built dialog corpora in two domains: weather information service and car navigation.

5.1 Real user evaluation

We built a dialog corpus in weather information service to measure the performance of the dialog system using our approach by real user evaluation. This corpus consists of 99 dialogs with 503 user utterances (turns). User's utterances were annotated with the semantic frame including speech acts, main goal and component slots for training the SLU module and indexing the DEDB.

To evaluate the preliminary performance, four test volunteers among computer science people evaluated our dialog system with five different weather information-seeking tasks. The volunteers typed their utterances with a keyboard rather than using a real ASR because it is hard to control the WER. We employed a simulated ASR error channel by generating random errors to evaluate the performance of dialog management under various levels of WER. We will explain the details of our ASR channel simulator in Section 5.2. The WER is controlled by this ASR channel simulator while the volunteers were interacting with computer. To measure the user perception of task completion rate (TCR), the volunteers evaluated the system's response in each dialog to measure the success turn rate (STR) and decided whether the entire dialog was successful or not. We evaluated the performance of our dialog system based on criteria outlined in (Litman and Pan, 2004) by measuring user satisfaction, which is defined with a linear combination of three measures: TCR, Mean Recognition Accuracy (MRA), and STR.

User Satisfaction = αTCR +βSTR + γMRA

In our evaluation, we set α, β and γ to 1/3, so that the maximum value of the user satisfaction is one.

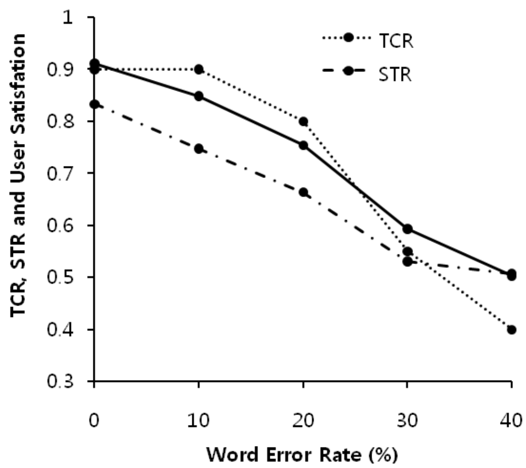

Figure 6 Dialog system performance with various word error rates in weather information seeking tasks. Dotted line is TCR; dashed line is STR; solid line is user satisfaction.

TCR, STR and user satisfaction decreased with WER. User satisfaction has relatively high value when the WER is smaller than 20% (Figure 6). If the WER is equal or over 20%, user satisfaction has small value because the TCR decreases rapidly in this range.

Generally, TCR has a higher value than STR, because although a dialog turn may fail, users still have a chance to use other expressions which can be well recognized by the system. As a result of this, even when some dialog turns fail, the task can be completed successfully.

TCR decreases rapidly when WER ≥20%. When WER is high, the probability of losing the information in a user utterance is also large. Especially, if words contain important meaning, i.e., values of component slots in SLU, it is difficult for the system to generate a proper response.

STR is 0.83 when WER is zero, i.e., although all user inputs are correctly recognized, the system sometimes didn't generate proper outputs. This failure can be caused by SLU errors or malfunction of the dialog manager. SLU errors can be propagated to the dialog manager, and this leads the system to generate a wrong response because SLU results are inputs of dialog manger.

If the WER is 20%, user satisfaction is relatively small because TCR decreases rapidly in this range. This means that our approach is useful in a system devoted to providing weather information, and is relatively robust to speech errors if the WER is less than 20%.

5.2 Simulated user evaluation

We built another dialog corpus in car navigation service to measure the performance of the dialog system by simulated user evaluation. This corpus consists of 123 dialogs with 510 user utterances (turns). The SLU result frame of this corpus has 7 types of speech acts, 8 types of main goals, and 5 different component slots.

The user simulator and ASR channel simulator has been used for evaluating the proposed dialog management framework. The user simulator has two components: an *Intention Simulator* and a *Surface Simulator*. The *Intention Simulator* generates the next user intention given current discourse context, and the *Surface Simulator* generates user sentence to express the generated intention.

ASR channel simulator simulates the speech recognition errors including substitution, deletion, and insertions errors. It uses the phoneme confusion matrix to estimate the probability distribution for error simulation. ASR channel simulator distorts the generated user utterance from Surface Simulator. By simulating user intentions, surface form of user sentence and ASR channel, we can test the robustness of the proposed dialog system in both speech recognition and speech understanding errors.

We defined a final state of dialog to automatically measure TCR of a simulated dialog. If a dialog flow reaches the final state, the evaluator regards that the dialog was successfully completed. TCRs and average dialog lengths were measured under various WER conditions that were generated by ASR channel simulator. Until the SLU result is an actual input of the dialog manager, we also measured the SLU accuracy. If a SLU result is same as a user's intention of the Intention Simulator, then the evaluator considers that the result is correct. Unlike in the real user evaluation, the dialog system could be evaluated with relatively large amount of simulated dialogs in the simulated user evaluation. 5000 simulated dialogs were generated for each WER condition.

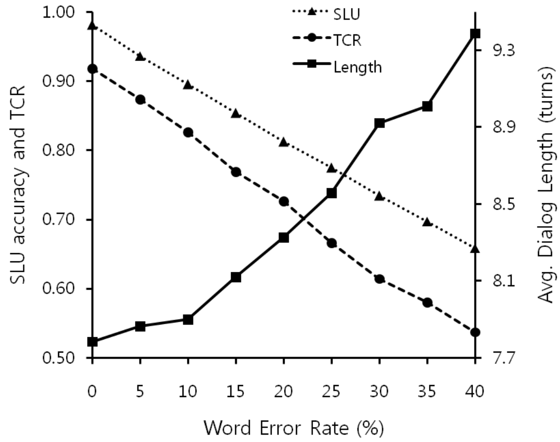

Figure 7 TCR, SLU accuracy, and average dialog length of the dialog system under various WER conditions.

We found that the SLU accuracy and TCR linearly decreased with the WER. Similar in the human evaluation, TCR is about 0.9 when WER is zero, and it becomes below 0.7 when WER is higher than 20%. Average dialog length, on contrary, increased with WER, and it has similar values when WER is less than 10% although it increased relatively rapidly when WER is higher than 15%.

6 Conclusions

This paper proposed a new probabilistic method to manage the human-machine dialog by using the frame-state belief state representation and the example-based system action hypothesis generation. The frame-based state representation reduces the computational complexity of belief update by grouping the indistinguishable user goal states. And the system generates the system action hypotheses with the example-based approach in order to refine the dialog flows easily. In addition, this approach employed the POMDP formalism to maintain belief distribution over dialog states so that the system can be robust to speech recognition errors by considering the uncertainty of user's input.

A prototype system using our approach has been implemented and evaluated by real and simulated user. According to the preliminary evaluation, our framework can be a useful approach to manage a spoken dialog system.

We plan to progress the research on adopting a formalized online search to determine the optimal system action (Ross and Chaib-draa, 2007). With the online searching, system doesn't need to behave the useless computation because this approach searches only possible path. We expect that this property of the online searching show the synergetic effect on dialog management if it combines with example-based approach.

Similar to example-based approach, the case-based reasoning approach (Eliasson, 2006) can be helpful for our future research. Some properties such as using previous cases to process current case can be shared with our approach. We think that some other properties including the concept of online learning can be useful for making our approach concrete

Acknowledgments

This research was supported by the MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute for Information Technology Advancement) (IITA-2008-C1090-0801-0045)

References

Changki Lee, Jihyun Eun, Minwoo Jeong, and Gary Geunbae Lee, Y. Hwang, M. Jang, "A multi-strategic concept-spotting approach for robust understanding of spoken Korean," *ETRI Journal*, vol. 29, No.2, pp. 179-188, 2007.

Cheongjae Lee, Sangkeun Jung, Jihyun Eun, Minwoo Jeong and Gary Geunbae Lee, "A situation-based dialogue management using dialogue examples," in *Proceedings of International conference on Acoustics, Speech, and Signal Processing*, Toulouse, 2006.

Diane J. Litman and Shimei Pan, "Empirically evaluating an adaptable spoken dialogue system," in *Proceedings of the 8th International Conference on Spoken Language Processing,* pp. 2145-2148, 2004.

Finale Doshi and Nicholas Roy, "Efficient Model Learning for Dialog Management," in *Proceeding of the ACM/IEEE international conference on Human-robot interaction*, Washington DC, 2007.

Jason D. Williams and Steve Young, "Scaling POMDPs for dialog management with composite summary point-based value iteration (CSPBVI)," in *Proceedings of AAAI Workshop on Statistical and Empirical Approaches for Spoken Dialogue Systems*, Boston, 2006.

Jason D. Williams and Steve Young, " Partially Observable Markov Decision Processes for Spoken Dialog Systems." *Computer Speech and Language* 21(2): 231-422, 2007

Karolina Eliasson, "The Use of Case-Based Reasoning in a Human-Robot Dialog System*", Licentiate of Engineering Thesis of Linköping Institute of Technology at Linköping University*, 2006

Nicholas Roy, Geoffrey Gordon, and Sebastian Thrun, "Finding approximate pomdp solutions through belief compression," *Journal of Artificial Intelligence Research*, vol. 23, pp.1–40, 2005.

Sptéphane Ross, Brahim Chaib-draa, "AEMS: An Anytime Online Search Algorithm for Approximate Policy Refinement in Large POMDPs", in *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2007

Steve Young, Jost Schatzmann, Karl Weilhammer and Hui Ye, "The hidden information state approach to dialog management," in *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, Honolulu, 2007.

Tim Paek and Eric Horvitz, "Optimizing automated call routing by integrating spoken dialog models with queuing models," in *Proceedings of HLT-NAACL*, pp. 41-48, Boston, 2004.