Available online at www.sciencedirect.com
ScienceDirect
Computer Speech and Language 24 (2010) 609–631
COMPUTER SPEECH AND LANGUAGE
www.elsevier.com/locate/csl

Hybrid approach to robust dialog management using agenda and dialog examples

Cheongjae Lee *, Sangkeun Jung, Kyungduk Kim, Gary Geunbae Lee

Department of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), San 31, Hyoja-Dong, Pohang 790-784, South Korea

Received 6 February 2009; received in revised form 10 June 2009; accepted 21 August 2009
Available online 6 September 2009

Abstract

Spoken dialog systems have difficulty selecting which action to take in a given situation because recognition and understanding errors are prevalent due to noise and unexpected inputs. To solve this problem, this paper presents a hybrid approach to improving robustness of the dialog manager by using agenda-based and example-based dialog modeling. This approach can exploit *n*-best hypotheses to determine the current dialog state in the dialog manager and keep track of the dialog state using a discourse interpretation algorithm based on an agenda graph and a focus stack. Given the agenda graph and multiple recognition hypotheses, the system can predict the next action to maximize multi-level score functions and trigger error recovery strategies to handle exceptional cases due to misunderstandings or unexpected focus shifts. The proposed method was tested by developing a spoken dialog system for a building guidance domain in an intelligent service robot. This system was then evaluated by simulated and real users. The experimental results show that our approach can effectively develop robust dialog management for spoken dialog systems.

Keywords: Example-based dialog modeling; Agenda-based dialog management; Robust dialog management; Error handling

1. Introduction

Spoken dialog systems communicate with human users using spoken language to accomplish a domain-specific task. For example, a user might use the spoken dialog system to reserve a flight over the phone, to direct a robot to guide him to a specific room, or to control in-car devices such as a music player and a navigator. Development of such spoken dialog systems involves human language technologies which must cooperate to answer user queries. Since the performance in human language technologies such as automatic speech recognition (ASR) and spoken language understanding (SLU) have been improved, spoken dialog systems can be developed now for many different application domains. Nevertheless, there are major problems for practical spoken dialog systems. One of them which must be considered by the dialog manager (DM) is the error

* Corresponding author. Tel.: +82 54 279 5581; fax: +82 54 279 2299.
E-mail address: lcj80@postech.ac.kr (C. Lee).

0885-2308/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.csl.2009.08.003

propagation from ASR and SLU modules. In general, errors in spoken dialog systems are prevalent due to errors in speech recognition or language understanding. These errors can cause the dialog system to misunderstand a user and in turn lead to an inappropriate response. To avoid these errors, a basic solution is to improve the accuracy and robustness of the recognition and understanding processes. However, it has been impossible to develop perfect ASR and SLU modules because of noisy environments and unexpected inputs. Error propagation is a more serious problem for novice users. Empirically, we have observed that expert users (e.g., system developers or experienced users) can operate the dialog systems with a low error rate. In contrast, novice users often suffer from more errors, which is one reason why it is difficult to develop practical dialog systems in the real world. Novice users do not usually know the limitations in vocabulary, grammar, and application domain serviced by the spoken dialog systems. Therefore, unexpected inputs are inevitable in the spoken dialog systems. However, dialog systems deployed in the real world should be broadly used by novices as well as experts. Therefore, the development of robust dialog management has also been one of the most important goals in research on practical spoken dialog systems.

Traditional DMs typically deal with these errors by adopting dialog mechanisms for detecting and repairing potential errors at the conversational level (Mctear et al., 2005; Torres et al., 2005; Lee et al., 2007). In human–computer communication, the goal of error recovery strategy is to maximize the user's satisfaction by guiding the repair of the wrong information by human–computer interaction. On the other hand, different approaches improve the robustness of dialog management using *n*-best hypotheses with uncertainty. Rather than Markov decision processes (MDPs) (Levin et al., 2000), partially observable MDPs (POMDPs) potentially provide a much more powerful framework for robust dialog management because they consider *n*-best hypotheses to estimate the distribution of the belief state (Williams and Young, 2007).

However, spoken dialog systems still commonly have ASR and SLU errors, so users still have difficulty successfully completing the tasks. To address this problem, we pursue a hybrid approach to robust dialog management using agenda and dialog examples. In general, the current dialog state is highly correlated to the previous dialog state during task-oriented dialogs. Therefore, when the current dialog state is estimated with uncertainty, the hierarchy and the order of the subtasks needed to complete the task should be reflected, usually as prior knowledge of the agenda. Furthermore, the agenda can help to recover ASR and SLU errors when unexpected states are detected because the system could know what the user should do to achieve his desired goal at the current state. In addition, practical dialog systems should be controlled by system developers based on business strategies which are implemented as hierarchical task structures and agendas. For example, payment for a flight reservation must be verified before the reservation is finally determined. Consequently, this paper proposes a new dialog management framework based on example-based dialog modeling (EBDM) framework (Lee et al., 2009) to improve robustness with a prior-knowledge agenda graph in which the hierarchy of the natural dialog flow is encoded from opening to closing dialogs for each domain-specific task. We consider the domain-specific agenda graph to estimate the best dialog state and system action by rescoring *n*-best recognition hypotheses and to give help messages given a focus stack when the user cannot continue the current dialog.

This paper is organized as follows. Section 2 reviews some related work as background, and Section 3 introduces the method and problems of the previous EBDM framework. Section 4 presents an agenda-based approach to combining prior knowledge. Following that, Section 5 explains greedy selection to rescore *n*-best hypotheses. Section 6 describes the error recovery strategies to handle unexpected cases. Then, Section 7 provides the experimental results of simulated and real user evaluations to verify our approach. Finally, Section 8 presents our conclusions and suggestions for future work.

2. Background

2.1. Robust dialog management

Robust dialog management has been a challenging topic because ASR and SLU modules are inevitably error-prone, so a DM must consider noisy inputs with uncertainty. There are many approaches to this problem such as the error handling and *n*-best exploiting approaches.

An error handling approach for spoken dialog systems involves a number of stages that include error detection and error recovery (Walker et al., 2000; Bohus and Rudnicky, 2005; Mctear et al., 2005; Torres et al., 2005; Jung et al., 2008). Several approaches have been proposed to detect and handle the errors generated in recognition and understanding processes. The most commonly used measure for error detection is a confidence score (Koo et al., 2001; Hazen et al., 2002; Lo and Soong, 2005). The decision to engage this method is typically based on comparing the confidence score against the manually preset threshold. However, confidence scores are not entirely reliable and are dependent on noisy environments and user types. In addition, false acceptance, where the confidence score exceeds the threshold but an error actually occurs, is more problematic as it may not be easy for the user to correct the system and put the dialog back on track. Thus, it can bring some problems at the level of dialog management.

The DM can adopt some error recovery strategies (e.g., explicit/implicit confirmation and rephrasing) to repair these errors (Skantze, 2005). An explicit confirmation takes the form of a question that asks explicitly for confirmation of the main slots of the task (e.g., "room name", "room type" and "room number" in the building guidance system). This may be accompanied by a request to answer with "yes" or "no". The DM can also use an implicit confirmation in which the system embeds in its next question a repetition of its understanding of what the user said in the response to the previous question. Explicit and implicit confirmations are good strategies to repair unreliable information by computing confidence scores on various levels including the phonetic level, the word level, and the utterance level. In these cases, the user says a partial phrase or a short utterance to acknowledge and confirm the dialog state. However, the deficiency of context may lead to new recognition and understanding errors. In addition, the distribution of user behaviors in coping with errors shows that users who achieve successful error recoveries use significantly more rephrasals than attempts to repair a chain of errors (Shin et al., 2002). For these reasons, we believe that the rephrase strategy repairs errors more successfully in the DM.

The traditional rephrase strategy of a DM component is often of little or no help, suggesting, "Please try using shorter sentences," or "I did not understand. Please rephrase your query." However, just repeating the previous utterance cannot always correct the recognition and understanding errors. In particular, novice users have potential out-of-vocabulary (OOV) and out-of-grammar (OOG) problems because they do not know the in-coverage vocabularies and grammars (utterance patterns). Considering novice users, the system should provide help messages to speak well-recognizable and well-understandable utterances at the current dialog state. Ideally, one of the best error recovery strategies is that the users can gradually learn how to operate the dialog system. Thus, a short tutorial allowed novice users to learn system limitations quickly (Kamm et al., 1998). However, this requires an additional session (i.e., it takes about a few minutes) before the use of the spoken dialog system, and it is not online help messages. Therefore, some groups have also investigated help generation mechanisms for novice users (Hockey et al., 2003; Fukubayashi et al., 2006). A spoken dialog system with immediate help messages was proposed to improve a dialog system performance for novice users (Hockey et al., 2003). This approach uses the grammar-based recognizer to detect out-of-coverage utterances and then generates the targeted help messages using in-coverage examples. Dynamic help generation was also developed by estimating the gap between user's mental model and the system (Fukubayashi et al., 2006). They represent the user's mental model on the domain concept tree in which each node has known degrees that denote the degree of how much a user understands the concepts corresponding to the nodes. The grammars and the domain concept trees are manually designed by system developers to recover ASR and SLU errors.

In addition, it is useful to exploit n-best recognition hypotheses rather than a single recognition hypothesis to determine the dialog state because the top recognition hypothesis is not always correct and the correct recognition hypothesis can have a lower rank in the n-best list. In the DM, the n-best hypotheses of the recognition and understanding modules are considered to estimate the belief state with the uncertainty in the framework of POMDPs (Williams and Young, 2007). These frameworks are statistically data-driven and provide theoretically principled dialog modeling to dynamically allow changes to the dialog strategy and to the actions of a dialog by optimizing some kinds of rewards or costs given the current dialog state with n-best recognition hypotheses. However, practical deployment of these approaches in dialog systems has some weaknesses (Paek, 2006). For example, the optimized policy may remove control from system developers because the optimization process in the classical POMDP framework is free to choose any action at any time. Therefore, incorporating domain knowledge or manually designed constraints is difficult for system developers to

have the opportunities to easily control the dialog flows in practical systems (e.g., it is obvious that the system should never print a ticket before it has asked for the origin city, but there is no direct way to communicate this to the optimization process.) Thus, recent studies have combined the robustness of the POMDP with the developer control afforded in conventional approaches (Heeman, 2007; Williams, 2008). We here propose a hybrid approach to dialog modeling in which *n*-best recognition hypotheses are rescored using prior knowledge (a knowledge-based approach) and dialog examples (a data-driven approach) to determine the next system action. In addition, error recover strategies can be easily implemented based on our approach. This provides novice users with useful messages to help continue the current dialog.

Recently, a new corpus-based approach to dialog management and automatic evaluation has been proposed which uses supervised learning techniques from human–human dialog corpora (Griol et al., 2008). This approach finds the next system action to maximize the posterior probability given the current dialog state, which is an abstract state representation to reduce the data sparseness problem. Although the statistical approach would be robust to ASR and SLU errors, there is not yet an obvious way to support prior knowledge and *n*-best recognition hypotheses. However, our approach can use both a human–human dialog corpus and handcrafted knowledge to improve its robustness for practical dialog systems.

2.2. Agenda-based approach to spoken dialog system

Although the data-driven approaches for dialog systems may be more robust and portable than traditional knowledge-based approaches, various knowledge sources are used in many spoken dialog systems that have been developed recently (Rich and Sidner, 1998; Bohus and Rudnicky, 2003; Roy and Subramaniam, 2006; Schatzmann et al., 2007; Young et al., 2007). These knowledge sources contain task model, domain model, and agendas which are usually domain-dependent. Task-oriented dialog systems use these powerful representations to segment large tasks into more reasonable subtasks. These are manually designed for various purposes including dialog modeling (Rich and Sidner, 1998; Bohus and Rudnicky, 2003), search space reduction (Young et al., 2007), domain knowledge (Roy and Subramaniam, 2006), and user simulation (Schatzmann et al., 2007). For example, Collagen (Rich and Sidner, 1998) uses an approximate representation of a partial SharedPlan composed of alternating act and plan recipe nodes for internal discourse state representation and discourse interpretation as a plan tree. In addition, a RavenClaw dialog management has been presented as an agenda-based architecture using hierarchical task decomposition and an expectation agenda (Bohus and Rudnicky, 2003). The domain-specific dialog control is represented in the *Dialog Task Specification* layer using a tree of dialog agents, with each agent handling a certain subtask of the dialog task. Recently, the problem of a large state space in POMDP framework has been solved by grouping states into partitions using user goal trees and ontology rules as heuristics (Young et al., 2007). The system developers can afford to integrate obvious domain properties into the spoken dialog systems by using the heuristics.

We are also interested in exploring algorithms that would integrate this knowledge source for users to achieve domain-specific goals. This paper uses an agenda graph whose hierarchy reflects the natural order of dialog control. This graph is used to both keep track of the dialog state and to select the best system action using multiple recognition hypotheses for augmenting the previous EBDM framework (Lee et al., 2009). Dynamic help generation was also adopted as an error recovery strategy that provides immediate help messages using the agenda graph and dialog examples (Lee et al., 2007). Our error recovery strategies can use the discourse information to provide an intelligent guidance based on the agenda graph, and the help delivered may reflect what the user was trying to achieve at the current turn. Our approach is closely connected with the EBDM framework. Thus, we first explain our basic dialog modeling method to manage task-oriented dialogs.

3. Example-based dialog modeling

Our approach is implemented based on EBDM framework, a data-driven dialog modeling. This section begins with a brief overview of the EBDM framework, which was inspired by example-based machine translation (EBMT) (Nagao, 1984), a translation system in which the source sentence can be translated using similar example fragments within a large parallel corpus, without knowledge of the language's structure. The idea of EBMT can be extended to determine the next system actions by finding similar dialog examples within an

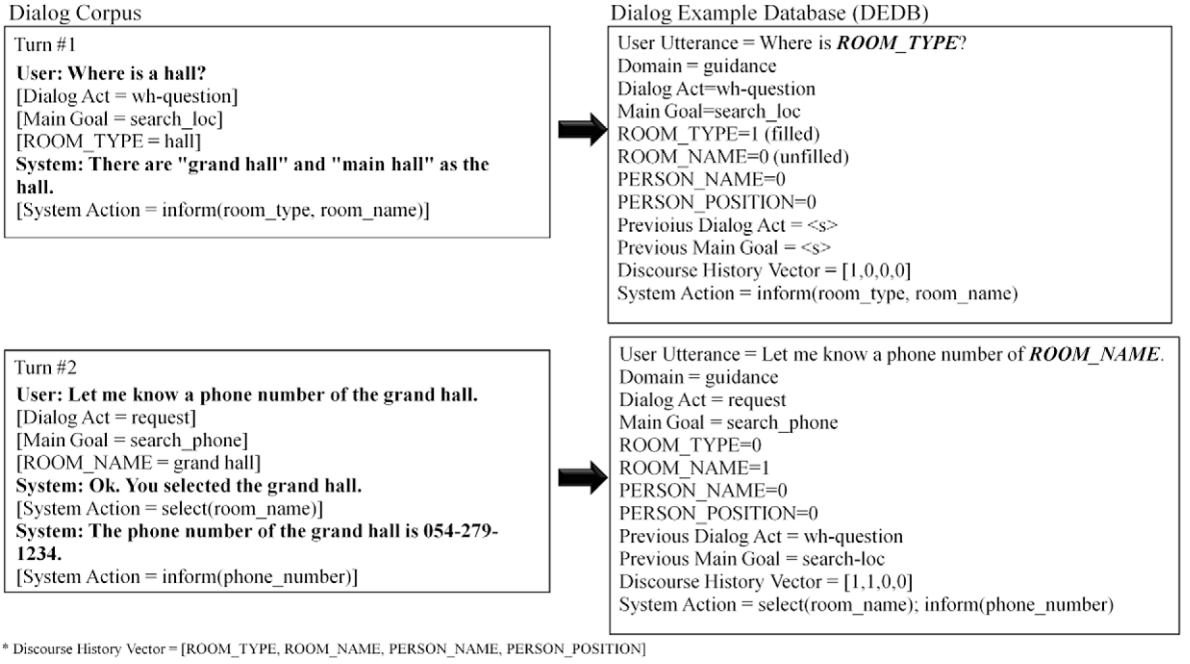


Fig. 1. Indexing scheme for dialog example database (DEDB) on a building guidance domain.

annotated dialog corpus. A dialog example is defined as a set of tuples that have the same semantic and discourse features. Each turn pair (one user turn and the corresponding system turns) in the dialog corpus is represented as one dialog example. The relevant examples are first grouped by a set of semantic and discourse features to represent the dialog state. The dialog examples are mapped into the relevant dialog state using a relational model that groups data using common attributes found in the data set because structured query languages (SQLs) can be easily manipulated to find and relax the dialog examples with some features. Then, the possible system actions are selected by finding semantically relevant user utterances with the current dialog state. The best system action can be expected to maximize a certain similarity metric.

A relational database is automatically built by first collecting a human–human dialog corpus related to predefined scenarios in each task. Then semantic tags (e.g., dialog act, main goal, and slot entity) are manually annotated to the user utterances, and system action tags to the system utterance. A hand-crafted automatic system is also used to extract discourse contextual features (e.g., previous intention and slot-filling status) by keeping track of the dialog states for each point in the dialog. After that, a dialog example database (DEDB) is semantically indexed to generalize the data; here the indexing keys can be determined according to state variables chosen by a system developer for domain-specific applications. Each turn pair (user turn, system turn) in the dialog corpus is mapped to semantic records in the DEDB (Fig. 1). The index constraints represent the state variables which are domain-independent attribute. Our basic constraints consist of general features to define the dialog state such as the current user intention (dialog act and main goal), slot flags, discourse history vector, and lexico-semantic string of the current utterance (Table 1).

To determine the next system action, the EBDM framework uses the three following processes:

- *Query generation:* DM generates a SQL statement using discourse history and the current dialog frame.[1]
- *Example search:* DM searches for semantically similar dialog examples in the DEDB given the current dialog state. If no example is retrieved, some features can be ignored by relaxing particular features according to the level of importance given the dialog's domain.

[1] In this paper, dialog frame denotes a kind of semantic frame representation. The SLU module predicts various kinds of semantic information (e.g., dialog act, main goal, and slots) from the current utterance.

Table 1
Index and search constraints for EBDM.

Constraints	Detail descriptions
Lexico-semantic string	Lexico-semantic patterns in which the values of domain-specific component slots on user utterance are replaced with their slot names (e.g. "meeting room" is replaced with LOC_ROOM_NAME)
Dialog Act (DA)	Domain-independent label of an utterance at the level of illocutionary force (e.g. STATEMENT, REQUEST, WH-QUESTION)
Main Goal (MG)	Domain-dependent user goal of an utterance (e.g. GUIDE-LOC, SEARCH-LOC, SEARCH-PHONE)
Slot Flag (SF)	Filling flag of the corresponding slot name at the current turn (e.g. If user says "meeting room" at the current turn, the flag of LOC_ROOM_NAME is set to 1)
Previous Dialog Act	Dialog act of the previous user utterance
Previous Main Goal	Main goal of the previous user utterance
Discourse History Vector	Binary vector for slot-filling status that is assigned a value of 1 if the component slot is already filled, and 0 otherwise

- *Example selection:* DM selects the best example to maximize the example score based on lexico-semantic similarity and discourse history similarity.

Fig. 2 illustrates the overall strategy of EBDM framework for the spoken dialog systems. The EBDM framework uses a robust statistical SLU approach in which semantic concepts are extracted from user's utterance by modeling using conditional random fields (CRFs) (Jeong and Lee, 2008). The content database (DB) contains several contents which denote a set of DB results (e.g., building information, person information) returned by the current dialog frame. The slot names and the slot values in the current dialog frame are transformed into a set of query constraints to find the user's desired contents. The discourse history stores the previous discourse information (e.g., the previous dialog frame, the previous contents, and the previous system action). This information has a stack structure of the previous discourse information from the dialog's start to the current turn. The previous discourse information represents the dialog state as the discourse features. The NLG module is based on system templates which allow a bit more flexibility with slots to be filled and the contents. The system templates are predefined according to each system action. Then, the NLG module translates the semantic representations of the system action to the spoken outputs in Korean. The concatenative TTS module (Kim, 2002) generates the speech output in Korean.

The EBDM framework is a simple and powerful approach to rapidly develop spoken dialog systems for multi-domain dialog processing (Lee et al., 2009). However, this framework must solve three problems for practical dialog systems for domain-specific tasks: (1) Keeping track of the dialog state to ensure steady pro-

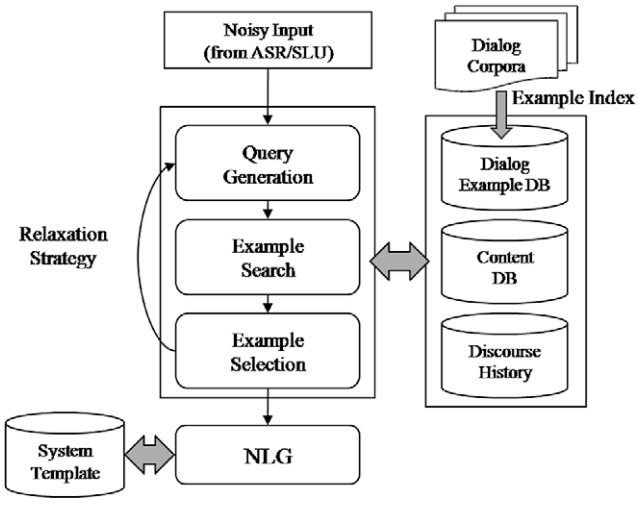


Fig. 2. Overall strategy of the EBDM framework.

gress towards task completion, (2) Supporting *n*-best recognition hypotheses to improve the robustness of dialog manager, and (3) Enabling error handling to recover ASR and SLU errors. Consequently, we sought to solve these problems by integrating the agenda graph as prior knowledge to reflect the natural hierarchy and order of subtasks needed to complete the task.

4. Agenda graph

In this paper, agenda graph G is simply a way of encoding the domain-specific dialog control to complete the task. An agenda is one of the subtask flows, which are possible paths from a root node to a terminal node. G is composed of nodes (v) which correspond to possible intermediate steps in the process of completing the specified task, and edges (e) which connect nodes. In other words, v corresponds to user goal states to achieve domain-specific subtask in its expected agenda. Each node includes three different components: (1) Preconditions that must be true before the subtask is executed; (2) A description of the node that includes its label and identifier; and (3) Links to nodes that will be executed at the subsequent turn. For every edge $e_{i,j} = (v_i, v_j)$ a transition probability is defined based on prior knowledge of dialog flows. This probability can be assigned based on empirical analysis of human–computer conversations, assuming that the users behave in consistent, goal-directed ways. Alternatively, it can be assigned manually at the discretion of the system developer to control the dialog flow. This knowledge has advantages for a practical spoken dialog system because a key condition for successful task-oriented dialog system is that the user and the system know which task or subtask is currently being executed.

For example, Fig. 3 illustrates part of the agenda graph for a building guidance domain used in a spoken dialog system. G is represented by a rooted directed acyclic graph (DAG), where each link in the graph reflects a transition between one user goal state and the next. The set of paths in G represents an agenda manually designed by the system developer. We adapted DAG representation because it is more intuitive and flexible than hierarchical tree representation. Although a tree is a connected acyclic graph, DAG representation makes it easier to reflect the dialog flows such as traditional finite state-based dialog modeling (Mctear, 1998). The syntax for graph representation in our system is described by an XML schema (Fig. 4).

4.1. Mapping examples to nodes

In the agenda graph G, each node v should hold relevant dialog examples corresponding to user goal states. Therefore, the dialog examples in DEDB are mapped to a user goal state when a precondition of the node is

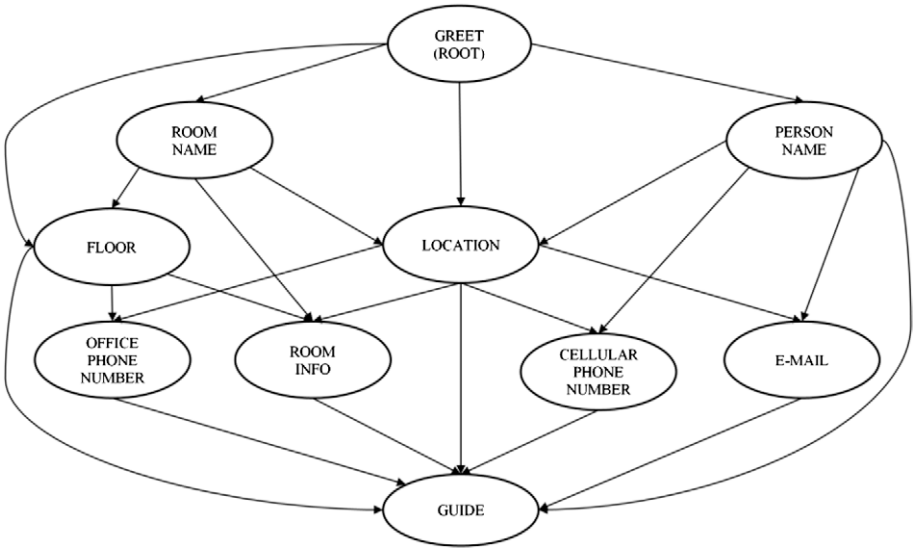


Fig. 3. Example of an agenda graph for a building guidance domain.

```
<agenda_graph domain="building_guidance">
 <node node_id="1">
   <property>                                                    NODE PROPERTY
     <label>Finding Room Name</label>
   </property>
   <precondition>
     <main_goal>SEARCH_LOC_NAME</main_goal>
     <current_slot name="LOC_ROOM_TYPE">1</current_slot>         PRECONDITION
   </precondition>
   <next>
     <node_id prob="0.25">3</node_id>   // To FLOOR
     <node_id prob="0.25">4</node_id> // To ROOM INFO            LINKS
     <node_id prob="0.5">5</node_id> // To LOCATION
   </next>
 </node>
</agenda_graph>
```

Fig. 4. AgendaXML description for the agenda graph.

true. Initially, the root node of the graph is the starting state. In general, the root node may be to prompt a system's greeting before using the spoken dialog system (e.g., "*SYSTEM: Hello. This is a building guidance system. How may I help you?*"). Then, prepared examples are mapped into each node, with the attributes of each dialog example examined via the preconditions of each node by breadth-first traversal from the root node. The preconditions are represented as a set of index and search constraints as described in Table 1. If the precondition is true, the node holds a relevant example that may appear in the user's goal state. How the DM selects the best of these examples will be described in Section 5.

4.2. Discourse interpretation

Inspired by Collagen (Rich and Sidner, 1998), we investigated a discourse interpretation algorithm that considers how the current user goal can contribute to the current agenda in a focus stack according to Lochbaum's discourse interpretation algorithm (Lochbaum, 1998). According to his work, each discourse event is explained as either starting a new task, continuing the current task, or completing the current task. In addition, the focus stack takes into account the discourse structure by keeping track of subtasks. In our system, the focus stack is a set of user goal nodes which lead to completion of the subtask. The top on the focus stack is the previous node in this set, and the focus stack is updated after every utterance.

These discourse events can be broken down into five main cases of possible current node for an observed user goal:

- ***NEW_TASK***: Starting a new task to complete a new agenda (Child of the root).
- ***NEW_SUB_TASK***: Starting a new subtask to partially shift focus (Sibling of the previous node).
- ***NEXT_TASK***: Working on the next subtask continuing the current agenda (Child of the previous node).
- ***CURRENT_TASK***: Repeating or modifying the observed goal on the current subtask (Current node).
- ***PARENT_TASK***: Modifying the observation on the previous subtask (Parent node).

Nodes in parentheses denote the topological position of the current node relative to the top node on the focus stack. If ***NEXT_TASK*** is selected, the current node is pushed to the focus stack. ***NEXT_TASK*** covers totally focused behavior (e.g. when there is no unexpected focus shift). This occurs when the current utterance is highly correlated to the previous system action. ***NEXT_TASK*** is illustrated by the example conversation shown in Fig. 5. In turn 1, a user wants to reserve a specific meeting room for a group meeting. Then, in turn 2, a robot informs the names of meeting rooms. After that, there are several possible subtasks to continue this dialog. One of them is that the user asks the room's capacity, focusing on the group meeting (in turn 3), or that the user asks the room's number to find its location (in turn 3′). Therefore, both turns correspond to ***NEXT_TASK*** cases in which the current turn is coherent to the previous turn. Nodes of ***NEXT_TASK*** case are positioned as children of the top node on the focus stack. The remaining four cases cover unexpected focus

Turn	Speaker	Utterance	SUBTASK
1	*U*	I am looking for meeting room to reserve group meeting.	ROOM NAME
2	*R*	There are 3 meeting rooms; Conference room, Meeting room A, and Meeting room B.	
3	*U*	How many persons are admitted to Conference room?	ROOM INFO
4	*R*	The Conference room can admit about 50 persons.	
3'	*U*	What is the room number?	LOCATION
4'	*R*	The room number of Conference room is 312	

Graph Example

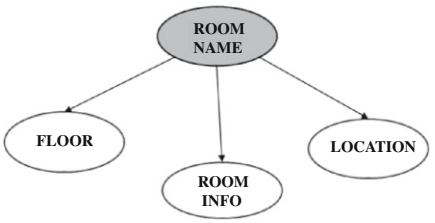


Fig. 5. Sample conversation with the agenda graph in the building guidance system. *U* indicates a user turn and *R* indicates a robot turn. Shaded node denotes the top node on the focus stack.

```
INTERPRET(<S,H>,G) ≡
  C←GENERATE(<S,H>,G)
  if |C| = 1
    then return discourse event in C
    else
      c* ← SELECT(<S,H>,C)
      return selected discourse event in c*
```

```
GENERATE(<S,H>,G) ≡
  return a union set of :
     i) NEW_TASK(<S,H>,G)
    ii) NEW_SUB_TASK(<S,H>,G)
   iii) NEXT_TASK (<S,H>,G)
   iv) CURRENT_TASK(<S,H>,G)
    v) PARENT_TASK(<S,H>,G)
```

```
NEXT_TASK(<S,H>,G) ≡
  C ←θ
  E ← retrieved examples of H
  foreach e∈E
    foreach c_s∈{children of top(S)|G}
      if e is an example of c_s
        then C ←C ∪ {c_s}
  return C
```

```
SELECT(<S,H>,C) ≡
  c*=argmax_c ωS_H(H)+(1-ω) S_D(c∈C|S)
  return c*
```

Fig. 6. Pseudo-codes for the discourse interpretation algorithm.

shift of discourse event. For example, ***NEW_SUB_TASK*** involves starting a new subtask to partially shift focus, thereby popping the previous goal off the focus stack and pushing a new user goal for the new subtask. ***NEW_TASK***, which is placed on the node linked to the root node, involves starting a new task to complete a new agenda. Therefore, the dialog is re-started and the current node is pushed onto the focus stack with the current user goal as its first element.

Fig. 6 shows some examples of pseudo-code used in the discourse interpretation algorithm to select the best node among possible next nodes.S, H and G denote the focus stack, recognition hypothesis, and agenda graph, respectively. The ***INTERPRET*** algorithm is initially called to interpret the current discourse event. Furthermore, the essence of a discourse interpretation algorithm is to find the candidate nodes C for the possible next subtask for an observed goal, expressed in the definition of ***GENERATE***. The ***SELECT*** algorithm selects the best node to maximize the score function based on current input and discourse structure given the focus stack. The details of how the score of candidate nodes are calculated are given in Section 5.

If none of the above five events holds, the discourse interpretation concludes that the current utterance should be rejected because we expect user goals to be correlated to the previous turn in a task-oriented domain. Therefore, this interpretation does not contribute to the current agenda on the focus stack due to ASR and SLU errors that often occur due to noisy environment and unexpected input. These cases can be handled by using the error recovery strategy in Section 6.

5. Greedy selection with *n*-best hypotheses

Many speech recognizers can generate a list of plausible hypotheses (*n*-best list) but output only the most probable one. Examination of the *n*-best list reveals that the best recognition hypothesis, the one with the lowest word error rate, is not always in top − 1 position but sometimes in the lower rank of the *n*-best list. Therefore, we need to select the recognition hypothesis that maximizes the scoring function among a set of

n-best hypothesis of each utterance. The role of the agenda graph is prior knowledge to rescore the n-best list by considering an agenda to successfully complete the task given a focus stack. The current system depends on a greedy policy which is based on immediate transitions rather than full transitions from the initial state. The greedy selection with n-best recognition hypotheses is implemented as follows. First every recognition hypothesis h_i is scanned and all possible nodes are generated using the discourse interpretation. Second scoring functions for a node score are computed for each candidate node c_k given a recognition hypothesis h_i. The greedy algorithm selects the node with the highest node score as the user goal state. Finally, the system actions are predicted by the dialog example to maximize the example score in the best node.

The generation of candidate nodes is based on multiple hypotheses from the previous EBDM framework. Although the previous EBDM chose a dialog example to maximize the example score, our system generates a set of multiple dialog examples with each score over a threshold given a specific recognition hypothesis. Then, the candidate nodes are generated by matching to each candidate example bound to the node. If the number of matching nodes is exactly one, that node is selected. Otherwise, the best node is selected by using multi-level score functions.

5.1. Node selection

The node selection is determined by calculating score functions. We defined multi-level score functions that range from 0.00 to 1.00 by combining some heuristic scores. The best node is selected by greedy searching with multiple hypotheses H and candidate nodes C as follows:

$$c^* = \arg\max_{h_i \in H, c_k \in C} \omega S_H(h_i) + (1-\omega) S_D(c_k|S) = \arg\max_{h_i \in H, c_k \in C} \lambda_{rec} S_{rec}(h_i) + \lambda_{cont} S_{cont}(h_i) + \lambda_{disc} S_{disc}(c_k|S),$$

where H is a list of n-best hypotheses and is a C set of nodes to be generated by the discourse interpretation. For the node selection, we divided the score function into two functions: $S_H(h_i)$, hypothesis score, and $S_D(c_k|S)$, discourse score, where c_k is a candidate node to be generated by single recognition hypothesis h_i.

We defined the hypothesis score at the utterance-level as:

$$S_H(h_i) = \lambda_{rec} S_{rec}(h_i) + \lambda_{cont} S_{cont}(h_i),$$

where $S_{rec}(h_i)$ denotes the recognition score which is a confidence score normalized over the score of the top-rank hypothesis. $S_{cont}(h_i)$ is the content score in the view of content management to access domain-specific contents. The score is defined as:

$$S_{cont}(h_i) = \begin{cases} 1 & \text{if } C_{h_i} \subseteq C_{prev}, \\ N(C_{h_i})/N(C_{total}) & \text{if } C_{h_i} \not\subset C_{prev}, \end{cases}$$

where C_{prev} is a set of contents at the previous turn and is C_{total} a set of total contents in the content database. C_{h_i} denotes a set of focused contents by recognition hypothesis h_i at the current turn. $N(C)$ represents the number of contents C. In Fig. 7, if a user has been focusing on the laboratories in the building at the previous turn (C_{prev}), then C_{h_1} and C_{h_2} (may be more probable contents than C_{h_3}) at the current turn because the contents are coherent to (or subsets of) the previous contents. This coherence means that the focus is not shifted so that the number of contents of interest is reduced by concrete slot information (e.g., the 3rd floor and AI laboratory). That is, this score can reflect the degree of content coherence because the number of contents of interest has been gradually reduced in task-oriented dialogs when there is no unexpected focus shift.

In addition to the hypothesis score $S_H(h_i)$, the discourse score $S_D(c_k|S)$ was defined at the discourse level to consider the discourse structure between the previous node and the current node given the focus stack S. This score is the degree to which candidate node c_k is in focus with respect to the previous user goal and system utterance. In the agenda graph G, each transition has its own probability as prior knowledge. Therefore, when c_k is ***NEXT_TASK***, the discourse score is easily computed as

$$S_D(c_k|S) = S_{disc}(c_k|S) = p(c_k|c = top(S)),$$

where $P(c_k|c = top(S))$ is a transition probability from the top node c on the focus stack S to the candidate node c_k. However, there is a problem for cases other than ***NEXT_TASK*** because the graph has no backward

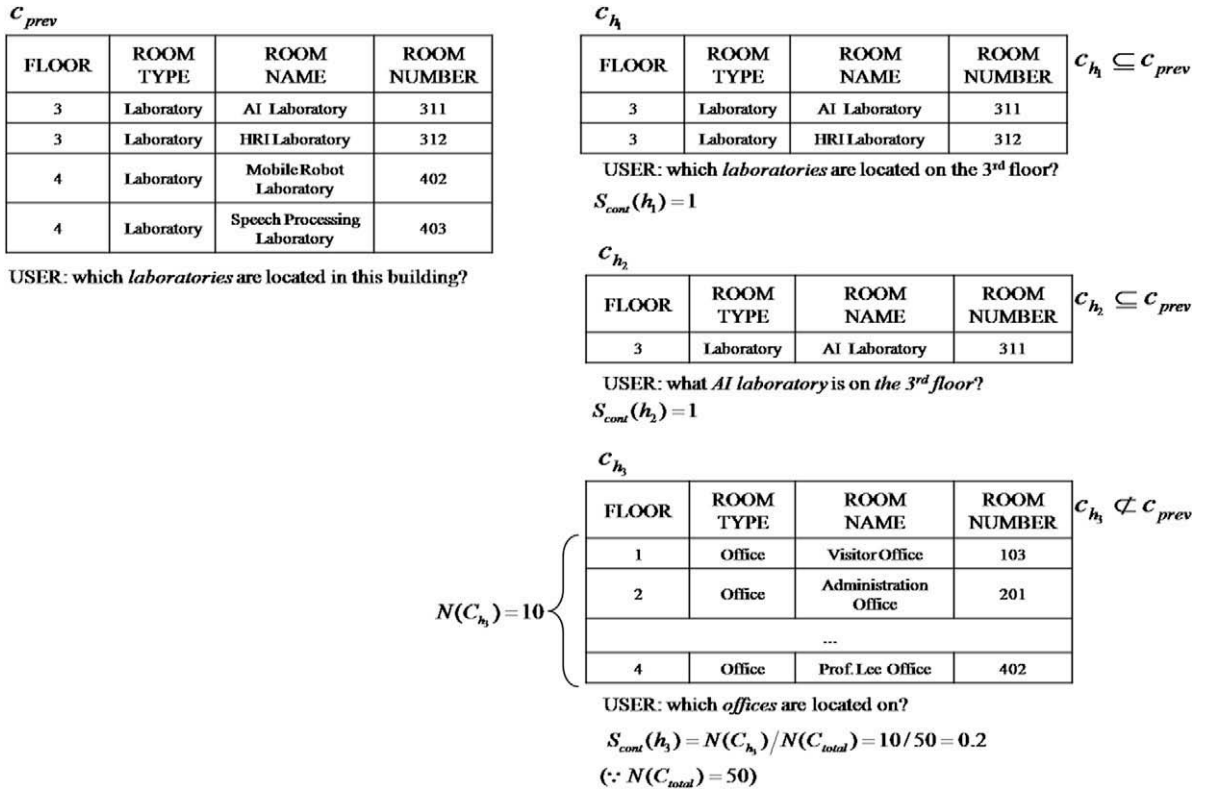


Fig. 7. Example of how to compute the content score.

probability. To solve this problem, we assume that the transition probability may be lower than that of the ***NEXT_TASK*** case because a user utterance is likely to be influenced by the previous turn. Actually, when using the task-oriented dialog systems, typical users stay focused most of the time during imperfect communication (Lesh et al., 2001). Therefore, the backward transition probability is assigned based on the minimum transition probability $p_{min}(S)$ from the top node on the focus stack to one of its children. Then the discourse score can be formalized when the candidate c_k node does not correspond to ***NEXT_TASK*** as follows:

$$S_{disc}(c_k|S) = [p_{min}(S) - \xi Dist(c_k, c)]_+,$$

where $[Z]_+ = \max(Z, 0)$ denotes some positive score having a distance penalty between the candidate node and the previous node,$Dist(c_k)$, according to type of candidate node such as ***NEW_TASK*** and ***NEW_SUB_TASK***. The simplest case is to uniformly assign ξ to a specific value (e.g. ξ). In Fig. 8, the user knew the location of a specific room at the previous turn. Although many subtasks can be done after knowing the location, the possible subtasks are limited by which subtasks correspond to each recognition hypothesis. In this case, the discourse score of ***NEXT_TASK*** (e.g., guiding the user to the room) is more probable than any other discourse event (e.g., asking someone's e-mail address). However, although other discourse events do not frequently appear, they may be selected in the real world (e.g., correcting the previous errors or restarting the dialog) so that the discourse scores of both ***PARENT_TASK*** and ***NEW_SUB_TASK*** are penalized by the distance penalty.

To select the best node using the node score, two empirical methods can be used to determine each weight. First, all weights can be determined based on the performance of ASR, SLU, and DM modules. The recognition, content, and discourse score functions can represent the scores of ASR, SLU, and DM modules, respectively. Thus, the score can be weighted if the source module yields relatively good performance. For example, if the ASR module yields the best performance, λ_{rec} could be weighted heavily. Second we can use $\omega(0 \leqslant \omega \leqslant 1)$ as an interpolation weight between the hypothesis score $S_H(h_i)$ and the discourse score $S_D(c_k|S)$. This weight is empirically assigned according to the characteristics of the dialog genre and task.

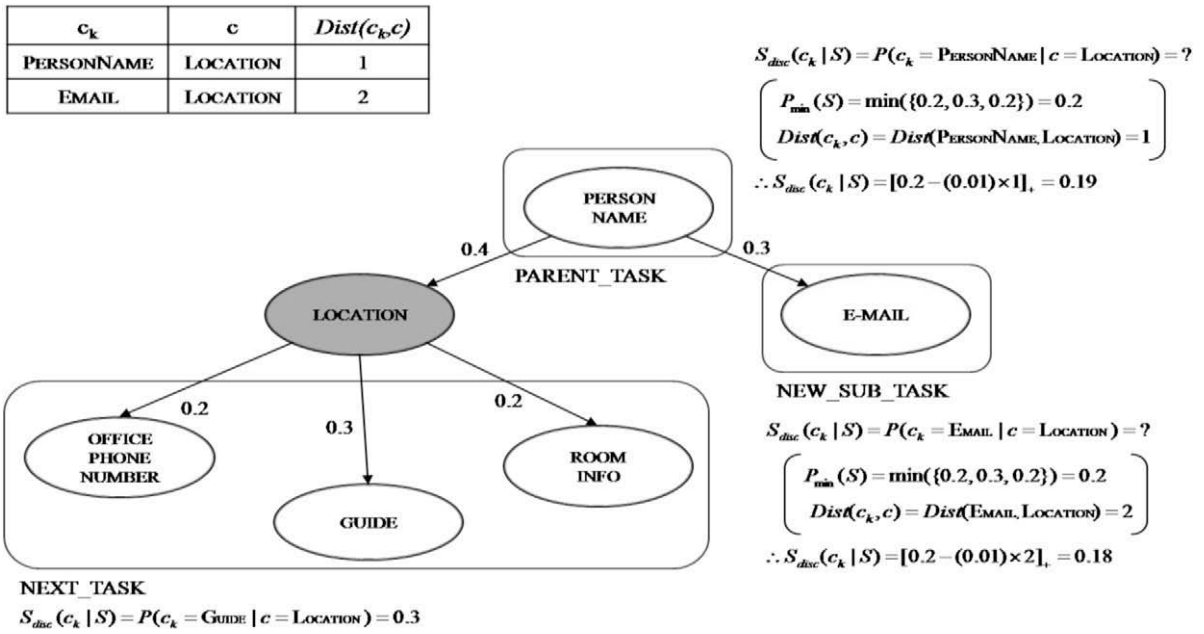


Fig. 8. Example of how to compute the discourse score. Shaded node denotes the top node on the focus stack.

For example,ω can be set lower to manage a transactional dialog in which the user utterance is highly correlated to the previous system utterance, e.g., a flight reservation task, because this task usually has a preferential order to fill slots. ω can be set higher for some tasks in which the current turn is weakly correlated with the previous turn, e.g., a weather information task, because the slots in this task are frequently changed so that the current turn may be more important than the previous turn to determine the system action. In this paper, although the weights could be empirically assigned, they were determined by using a simple grid search method.

5.2. Example selection

After selecting the best node c^* and the best recognition hypothesis h^*, the example score is used to select the best dialog example among a set of the candidate examples $E(c^*)$ mapped into the best node:

$$e^* = \arg\max_{e_i \in E(c^*)} S_e(h^*, e_i) = \arg\max_{e_i \in E(c^*)} \lambda_{lex} S_{lex}(h^*, e_i) + \lambda_{dhv} S_{dhv}(h^*, e_i) + \lambda_{sem} S_{sem}(h^*, e_i),$$

where $\lambda_{lex}, \lambda_{dhv}$ and λ_{sem} are the weights of lexico-semantic similarity, discourse history vector similarity, and semantic similarity, and the sum of the weights should be one.

The lexico-semantic similarity is defined as a normalized edit distance between lexico-semantic patterns of the best recognition hypothesis and the example's lexico-semantic pattern. In normalized edit distance, a cost function $C(i, j)$ is defined as

$$C(i,j) = \begin{cases} 0 & \text{if } w_{h,i} = w_{e,j}, \\ 0.5 & \text{if } w_{h,i} \neq w_{e,j} \text{ and } w_{e,j} \notin S_{slot}, \\ 1 & \text{if } w_{h,i} \neq w_{e,j} \text{ and } w_{e,j} \in S_{slot}, \end{cases}$$

where $w_{h,i}$ denotes ith word of the user's utterance (ASR hypothesis), $w_{e,j}$ denotes jth word of the user's utterance in the dialog example e, and S_{slot} is a set of slot names (Table 2). Some words in the user's utterance represented as a lexico-semantic pattern contain specific slot names. For example, in the utterance "USER: where is *meeting room*", '*meeting room*' is a slot having important information to find appropriate contents. The SLU module can identify the slot so that the DM knows that the word of 'meeting room' (slot value) is 'LO-

Table 2
List of slots used in the building guidance domain.

Slot name	Description	Examples of slot value
LOC_ROOM_NAME	The name of a room in a building	President room, AI laboratory, Conference room
LOC_ROOM_NUMBER	A number of a room in a building	Room No. 101, No. 102, 301
LOC_ROOM_TYPE	More general category of a room having particular features in common	Meeting room, Laboratory, Office
LOC_FLOOR	A floor of a building on a particular level	The 1st floor, The 2nd floor, The 3rd floor
PER_NAME	The name of the person working (or residing) in the building	Tom Smith, Cheongjae Lee, Gary Lee
PER_TITLE	A position (or role) of someone in a company or organization	President, Professor, Researcher

C_ROOM_TYPE' (slot name). Therefore, when the user just says "where is *meeting room*," the DM can convert it into a lexico-semantic pattern in which the slot value is replaced with its slot name (e.g., in the utterance "USER: where is*[LOC_ROOM_TYPE]?*", the square bracket indicates the slot name.) We assign a higher cost when the slot name on the example's utterance aligns to different word or slot name because the slot names (e.g., '*[LOC_ROOM_TYPE]*', '*[LOC_ROOM_NAME]*') are more important than other words (e.g., 'a/an', 'the', 'is', 'there') in the task-oriented dialog systems. This similarity reflects which example is lexically closer to the current utterance. This is important for the detection of erroneous utterances containing substitution, deletion, and insertion errors from an imperfect ASR module.

An additional measure to reflect discourse information was also considered. The degree of the discourse history vector similarity is defined as follows:

$$S_{dhv}(h,e) \simeq S_{dhv}(v_h, v_e) = \frac{v_h \cdot v_e}{\|v_h\|\|v_e\|} = \frac{\#\text{ of slot-filling status in common}}{\#\text{ of component slots}},$$

where v_h is the discourse history vector of the current dialog state, v_e is the discourse history vector of the example's dialog state, and $\|\cdot\|$ denotes a Euclidean norm. This similarity describes a cosine measure between v_h and v_e and is a widely used measure of vector representation. These vectors are assigned a value of 1 if the component slot is already filled during the dialog, and 0 otherwise. This measure is used to compare the discourse

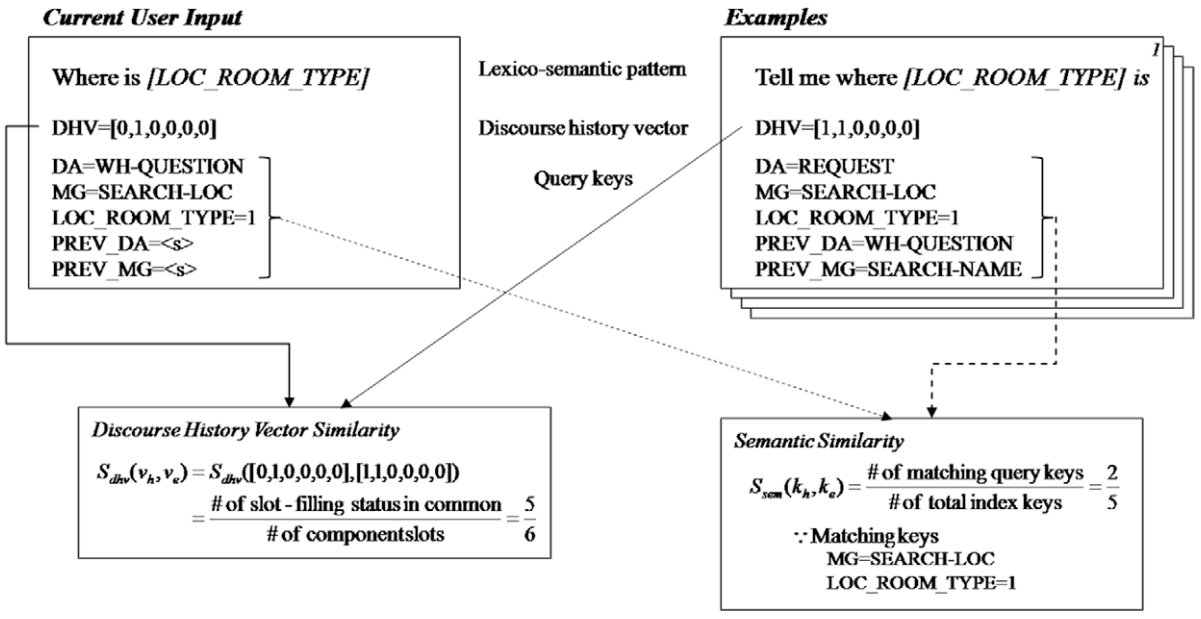


Fig. 9. Example of how to compute the discourse history vector similarity and semantic similarity.

history of the current dialog state and the example's dialog state. This similarity indicates that the dialog state of a dialog example is more relevant to the current dialog state if the slot-filling status is more similar.

Finally, the semantic similarity is defined for example selection between a set of index keys for the current hypothesis k_h and that of the examples k_e,

$$S_{sem}(h,e) \simeq S_{sem}(k_h,k_e) = \frac{\#\text{ of matching query keys}}{\#\text{ of total index keys}}.$$

The semantic similarity is the ratio of matching query keys to the number of total index keys between the current hypothesis and the example records. This similarity reflects that a dialog example is semantically closer to the current utterance if the example is selected with more matching keys. Fig. 9 illustrates an example of how to compute the discourse history vector similarity and semantic similarity. If a user says "where is *meeting room*", its lexico-semantic pattern is "where is *[LOC_ROOM_TYPE]*". In this case, the discourse history vector similarity is computed as the cosine similarity between the current discourse history vector (e.g., only the slot of LOC_ROOM_TYPE is filled) and the discourse history vector of the candidate examples. The semantic similarity is computed as the matching ratio of the current query keys (e.g., dialog act, main goal, slot flags) to the indexed keys of the candidate examples.

After processing the node and example selection, the best example is used to predict the system actions. Then the template-based natural language generation module can generate surface-level system utterances using system actions and retrieved contents.

6. Error recovery strategy

Our approach assumes that the user behaves optimally and consistently because typical users stay focused most of the time during imperfect communication when using spoken dialog systems (Lesh et al., 2001). However, non-expert users of the spoken dialog systems usually produce unexpected inputs that are out-of-coverage patterns; when they do, recognition and understanding errors frequently occur. In addition, noisy environments can cause fatal mistakes in ASR and SLU modules so that the DM must handle out-of-coverage states. This situation usually contains potential errors because the utterance may be semantically or grammatically incorrect. In general, the cause of wrong system responses may lie in out-of-coverage problem and data sparseness problem of ASR, SLU, and DM models. In fact, it is impossible to cover all possible situations using a fixed number of dialog examples. This is serious problem in our framework, which can cause either *NoAgenda* or *NoExample* errors. To recover these errors, DM detects that something is wrong in the current input and takes immediate steps to address the problem using the *AgendaHelp* and *UtterHelp* error recovery strategies.

6.1. AgendaHelp strategy

As noted in Section 4.2, discourse interpretation sometimes fails to generate candidate nodes due to prior errors or unexpected focus shift. To complete the current agenda, the discourse events should belong to either

Turn	*Speaker*	*Utterance*	*SUBTASK*
1	*U*	Where is Conference room?	
2	*R*	Conference room is opposite the elevator on the 2nd floor.	LOCATION
3	*U*	Where the e-mail is? [misrecognized]	*NoAgenda*
4	*R*	Next, you can do the subtask (1) Asking the office phone number of the room, (2) Asking the room's information, or (3) Request to guide to the room	*AgendaHelp*

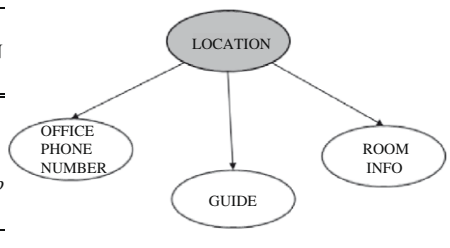


Fig. 10. Sample conversation for the *AgendaHelp* recovery strategy. Shaded node denotes the top node on the focus stack. In turn 3, the user does not know what to say at the current turn. Then, the system could give a help message for the next subtask to be done given the agenda graph and the focus stack.

NEW_TASK, ***NEW_SUB_TASK***, ***NEXT_TASK***, ***CURRENT_TASK***, or ***PARENT_TASK***. However, *NoAgenda* error occurs when none of the above five events holds at the discourse interpretation stage. In general, the current user intention may be highly related to the previous system action because the users in task-oriented dialogs behave in consistent and goal-directed ways to complete the current agenda. Therefore, it may be useful for the DM to provide a help message for what the user could do at the current dialog state (Fig. 10). For example, the DM for the building guidance domain could give a help message such as "*SYSTEM: Next, you can do the subtask: (1) Asking the office phone number of the room, (2) Asking the room's information, or (3) Request to guide to the room*" if the user does not know what to do after finding the room's location. The *AgendaHelp* recovery strategy can be generated by keeping track of the focus stack given a domain-specific agenda graph because the agenda-based approach can know the next possible subtasks to be achieved for the final goal. In the agenda graph, each node label indicates an abstract description for the corresponding subtask. Therefore, the DM can make a help message when the discourse interpretation could not generate any candidate node, which may be caused by either the user not knowing what to say or a prior error occurring.

6.2. UtterHelp strategy

We also define a *NoExample* error when no dialog examples are retrieved, rejecting the current input. *NoExample* means that the dialog system cannot find similar examples because unexpected inputs may be out-of-coverage patterns so that its example score falls below a predefined threshold at the example selection step. When the current input has a low example score, it can be rejected to reduce the risk of entering incorrect information into the dialog state. To address this error, an *UtterHelp* error recovery strategy is defined in which the DM gives an in-coverage example of what the user could say to complete the current subtask. The DM provides the in-coverage utterance template in DEDB because the examples are already trained to build the ASR, SLU, and DM modules. Thus, the on-line performances of ASR, SLU, and DM modules are increased by using these templates. The DM should handle a *NoExample* error by selecting the best template considering both the discourse context and the current input because the current input could contain correct observations even if containing some errors. First, the DM generates candidate nodes belonging to possible node transitions from the top node on a focus stack by the discourse interpretation. Then it selects the best example on the candidate nodes by calculating its example scores. For example, if there are no examples to ask the room's location, the DM could give a help message such as "*SYSTEM: I am sorry that I can't understand your input. You can rephrase using the similar expression – I am looking for [ROOM_NAME].*" We believe that the *UtterHelp* recovery strategy is more effective feedback than the typical rephrasing feedback such as "*SYSTEM: I am sorry that I can't understand your input. Please say it again.*"

7. Experiments and results

7.1. Testing setup

Experiments were conducted with a spoken dialog system developed for the building guidance domain in a intelligent service robot, PHOPE (Fig. 11) (Oh et al., 2009). The task in this domain is to provide information and navigation for rooms and persons in a public building (e.g., research institute, post office, city hall). Users may request specific values for six different slots: room name, room type, room number, floor, person name, and person title (e.g., president, director, general manager, assistant manager, and engineer). They may also then ask for information about the location details, such as office phone number, cellular phone number, and e-mail address. Finally, if the user selects a specific room to visit, then the robot can take the user to the desired room. A dialog corpus on building guidance domain was built using ten people who knew how to operate the dialog system. They collected text-based human–human dialogs including about 900 user utterances in 200 dialogs based on the Wizard-Of-Oz method (Table 3). We gave them a set of pre-defined subjects related to domain-specific tasks (e.g., Find the president's room for a visit). These human–human dialogs contain only reasonable examples because it is useless to manage task-oriented dialogs if incorrect examples are selected. Thus, the dialog flows should actually appear in the real world situations. In addition, the utterance patterns should be grammatically correct and the vocabulary should be typical of that used by the real users. All

Fig. 11. Intelligent robot of PHOPE for a building guidance.

Table 3
Dialog corpus statistics. DA: dialog act, MG: main goal, SA: system action.

# Dialogs	# UserUtterances	# Words	# DA	# MG	# Slot	# SA
214	879 (4.11 user's utters/dialog)	5848 (6.65 words/utter)	9	15	6	16

utterances were annotated for a supervised SLU module given a tag set of dialog acts, main goals, and domain-specific component slots. We also annotated the system actions manually for a DM module. Subsequently, this corpus was used to train user simulator, SLU, and dialog models. We also manually designed a domain-specific agenda graph based on the dialog corpus and prior knowledge (Fig. 3). A dialog system was built by employing three different methods to process the user utterances: P-E (examples only), P-ER (examples + recovery), P-EA (examples + agenda), and P-EAR (examples + agenda + recovery). The first baseline system (P-E) was based on the previous EBDM framework using only dialog examples. Note that this system can only make use of one recognition hypothesis at a time so any additional information regarding the uncertainty is lost. The next system (P-ER) uses the *UtterHelp* recovery strategy to handle exceptional cases such as *NoExample* error. This system provided a longer help message including what the user could say at this point. In these systems (P-E and P-ER), the current turn depended on the previous turn to search for the best example because there was no knowledge to reflect the agenda. The third (P-EA) and fourth systems (P-EAR) implemented our approach of using dialog examples and the agenda graph. P-EA system can support *n*-best ASR hypotheses to determine the best dialog state with the agenda graph. However, no error strategies were triggered even if the discourse interpretation or example selection failed. The P-EAR system can support both *n*-best ASR hypotheses and error recovery strategies such as both the *AgendaHelp* and *UtterHelp* error recovery strategies. All parameters (6 weights and 1 threshold) to compute multi-level score functions were determined by a grid search method. The grid search method can be used to find the grid to maximize the average score using the development set (100 simulated dialogs per grid).

In addition, to evaluate our dialog systems by simulating a realistic scenario, a simulated environment was developed with a user simulator and an ASR channel (Fig. 12) (Jung et al., 2009). The statistical user simulator is composed of two components: an *Intention Simulator* and a *Utterance Simulator*. First the *Intention Simulator* was implemented by using CRFs model to consider the dialog sequences as a sequential labeling problem. The simulator can calculate the conditional probability of the next user intention given the current discourse context features (e.g., previous system action and discourse history vector). Diverse user intentions were generated by randomly selecting the user intention based on the probability distribution of the user intention given the discourse context. After the user intention was selected, the *Utterance Simulator* generated a word-level user sentence to express the selected intention using intention-specific POS tag sequences and dictionaries. The ASR channel allows automatic transformation of the raw utterance to a noisy utterance at a fixed word error rate (WER). This channel used hand-crafted probability distributions to simulate speech recognition errors including substitution, deletion, and insertion. The substitution candidates were sampled by phoneme-level sequence alignment using the phoneme confusion matrix. Finally, the *n*-best ASR hypotheses

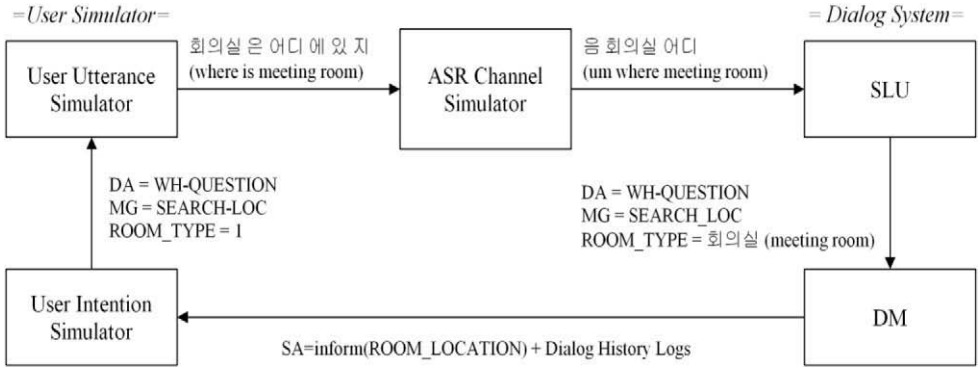


Fig. 12. Simulated environment for spoken dialog system evaluation.

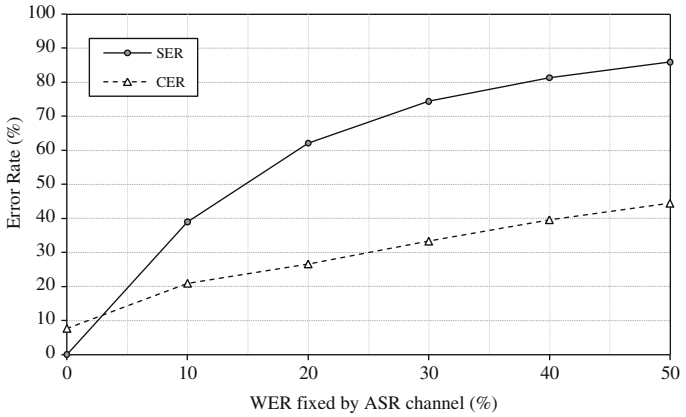


Fig. 13. Sentence error rate (SER) and concept error rate (CER) under various WER conditions.

were selected by rescoring a language model score among sampled noisy utterances because the distribution of language model score is independent of the noisy environments and the user types. Our ASR channel was verified by examining the correlation between WER and concept error rate (CER) (Fig. 13). Note that increasing WER increases the CER. This result shows that our ASR channel can model speech recognition errors as well as SLU errors at a certain error rate.

7.2. Evaluation metric

The success of a dialog was automatically evaluated by taking the final dialog state of the task completion (e.g., in the slot-filling task, the component slots are fully filled to one desired location, and the dialog manager generates corresponding system actions and utterances). This was used to compute Task Success Rate (TCR) and Average Turn Length (ATL). If a dialog flow reaches the final dialog state, the evaluator regards the dialog's task as successfully completed. The maximum turn length was set to 20 user turns per dialog because users are likely to halt the use of the dialog system.[2] In addition, the overall performance of the dialog systems was evaluated by defining an average score function similar to the reward score commonly used in RL-based dialog systems for measuring both a cost and task success. We give 20 points for the final dialog state and penalize 1 point for each action performed by the system to penalize longer dialogs. The maximum possible score is 18 (e.g., 20 minus 2 system actions: select a desired room directly in one turn, then navigate to the

[2] In fact, there are some users who may halt at 15, 25, or 30 turns. It depends on the individual user. However, we believe that the maximum turn length does not affect the relative performance significantly to compare each method.

room) and the minimum is −20 (e.g., the task is not yet completed after 20 system actions). We hypothesized that: (1) Rescoring *n*-best hypotheses would increase the average score (TCR would be increased and ATL would be reduced); and (2) Providing the help messages would improve user's ability to complete tasks.

7.3. Simulated user evaluation

First, our overall aim is to assess whether our approach provides an improvement in dialog performance at higher noise levels. We tested the robustness of the different dialog systems implemented according to WER values (Fig. 14). The average scores for the P-E, P-ER, P-EA, and P-EAR systems were measured under WER conditions ranging from 0% to 50% because an average WER of 20% (and up to 50% for novice users) are quite common for the traditional spoken dialog systems. The user simulator allowed the evaluation of the dialog system with a relatively large number of simulated dialogs, whereas the real user evaluation allowed the evaluation of only a few dialogs. A thousand simulated dialogs were evaluated at each WER condition. The average score decreases consistently as WER increases for all systems, but both the P-EA and P-EAR systems using 5-best ASR hypotheses attain larger scores than the P-E and P-ER system using only 1-best ASR hypothesis at all WER values. This result demonstrates that our approach successfully models dialog strategies and that it can tolerate speech recognition errors to a certain extent, although an increasing error rate results in a longer task completion time. In addition, the error recovery strategies could reduce user difficulties caused by the ASR errors or the lack of knowledge about the dialog flows. Thus, we conclude that the use of the *n*-best hypotheses with the agenda graph using error recovery (the P-EAR system) is helpful to implement a robust dialog management.

Next, we explored the effect of *n*-best size for robust dialog management (Fig. 15). The ASR channel generated *n*-best hypotheses at a fixed WER. In the user simulator, the generated *n*-best lists were pruned to a maximum of 100 hypotheses by rescoring a language model score. Generally, the P-EAR system achieved similar performances when no ASR errors occur, but supporting *n*-best ASR hypotheses ($n \geqslant 2$) outperforms the baseline system ($n = 1$) consistently for a range of WERs. This result illustrates that all *n*-best lists ($n \geqslant 2$) can increase the robustness relative to a single recognition hypothesis when some recognition errors occur, and about 5-to-10 best hypotheses provided the best performance in our dialog simulation.

In addition, we explored how many dialog examples are necessary to develop the spoken dialog system for the building guidance task (Fig. 16). When more examples were used to build the DEDB and were mapped into nodes on the agenda graph, the average scores were slightly increased, and saturating at about 800 examples. The average score of the P-EAR system was always larger than that of the P-E system irrespective of the number of dialog examples. This indicates that the approximately 900 examples used in our approach were enough to test different methods (e.g., P-EAR system outperforms P-E system when each system holds the same examples).

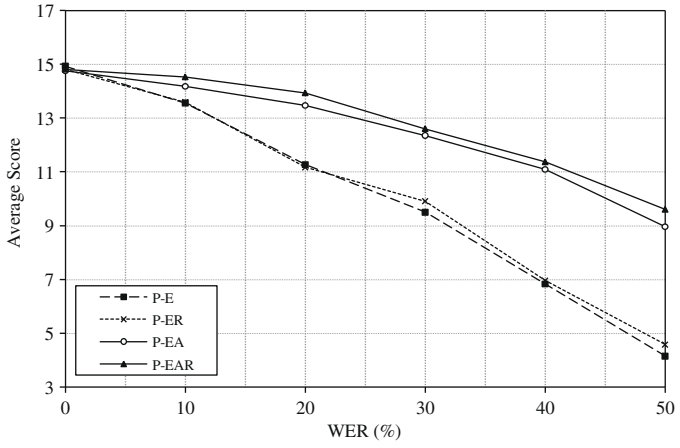


Fig. 14. The average scores for the P-EA, P-EAR systems using 5-best ASR hypotheses and the P-E, P-ER systems using 1-best ASR hypotheses. Note that each system is abbreviated as P(PHOPE), E(EBDM), A(AGENDA GRAPH), and R(RECOVERY).

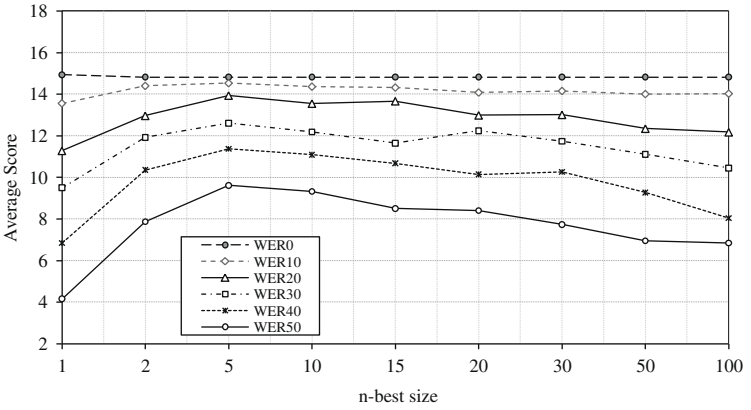


Fig. 15. The average score of the P-EAR system according to *n*-best size.

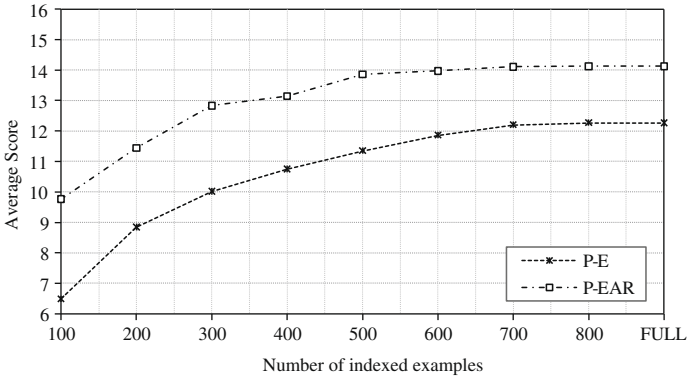
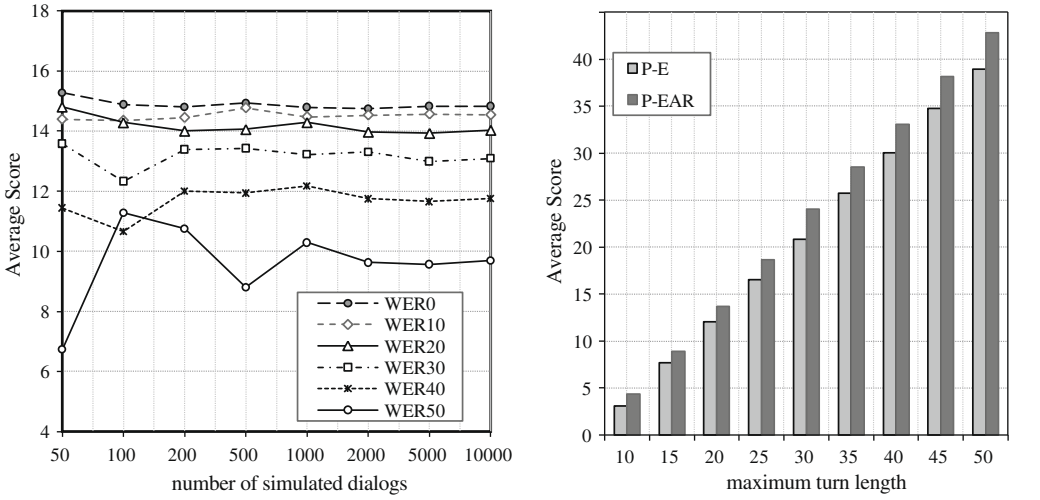


Fig. 16. The average score of the P-E and P-EAR systems according to the number of dialog examples used on the agenda graph (WER = 20%).

Finally, we also investigated the performance trend of the number of simulated dialogs and the maximum turn length (Fig. 17). When we evaluated the P-EAR system by using the dialog simulator, the use of over 1000 simulated dialogs had stable performances at each WER level (Fig. 17a). This means that 1000 simulated dialogs are enough to verify our approach. In addition, setting the maximum turn length to larger values yields larger average scores, because the probability of task completion was increased by repeated trials. However, the relative performances according to the maximum turn length shows the similar trends (e.g., the average score of P-EAR system was larger than that of P-E system at the same maximum turn length) (Fig. 17b).

7.4. Real user evaluation

In an attempt to quantify the impact of our approach in the real world, five Korean undergraduate students participated in a real user evaluation. They were provided 10 pre-defined tasks and asked to use the system to complete them. The 50 resulting dialogs contain about 150 utterances. Afterwards, the participants judged the task success of each dialog to assess the dialog performance evaluation. The speech recognition hypotheses were obtained by using Hidden Markov model Toolkit (HTK) speech recognizer. The acoustic model was trained with a Korean read speech database (about 70 h) from newspaper and textbooks (Lee and Chung, 2007). The language model was adapted to the building guidance domain using the dialog corpus. The lattice was constructed by a bigram model, and the *n*-best list on the lattice was rescored using a trigram model. The WER of our ASR module adapted to our application domain was about 21.03%. The results are shown in Table 4. The effect of our approach with *n*-best hypotheses was compared to the previous EBDM framework



(a) Average score vs. number of simulated dialogs (P-EAR) (b) Average score vs. maximum turn length (WER=20%)

Fig. 17. The average score according to the number of simulated dialogs and the maximum turn length.

Table 4
Real user evaluation.

System	ATL (user turns per dialog)	TCR (%)	AverageScore
P-E	4.65	84.0	12.15
P-EAR	4.35	90.0	14.52

which has no agenda graph and supports only the 1-best recognition hypothesis. Note that using 10-best hypotheses and the agenda graph increases the TCR from 84.0% to 90.0%, that is, 45 out of 50 dialogs were completed successfully, and the ATL was shorter, which shows 4.35 turns per a dialog. These results show that the use of the *n*-best hypotheses with the agenda graph helps to improve the robustness of the EBDM framework against noisy inputs.

8. Discussion and future work

This paper presented a hybrid approach to the robust dialog management using both data-driven and agenda-based approach. We proposed an effective dialog management technique to predict the next system action and to recover ASR and SLU errors by using dialog examples and agenda graph. The following briefly discusses its strengths, our experiences and future work.

8.1. Hybrid approach to robust dialog management

Our method of robust dialog management requires two resources: dialog examples (collected from a human–human dialog corpus as data), and the agenda graph (designed manually as prior knowledge). Our system can explicitly determine whether the current input is correct or not by comparing it to dialog examples. We believe that it is more useful to manage in-coverage inputs such as similar patterns and possible dialog states in practical systems because out-of-coverage inputs have potential errors. In fact, the agenda graph can be thought of as a hidden cost of applying our method. However, an explicit agenda is necessary to successfully achieve the purpose of using the spoken dialog systems. For example, the dialog manager can predict the next subtasks likely to be executed by the users. Then the dialog systems can load subtask-specific models such as acoustic, language, and understanding models to increase the performance of ASR and SLU modules. In industry, sys-

tem developers have an opportunity to control the dialog plans according to commercial needs for business because the agenda graph is created manually to encode their knowledge of the task and business rules.

With the dialog examples and the agenda graph, our approach allows the dialog system to rescore *n*-best recognition hypotheses with utterance-level and discourse-level scores. This can facilitate consideration of multiple hypotheses rather than a single recognition hypothesis to determine the best dialog state relying on contextual information. Our approach also incorporates some heuristics with hand-crafted knowledge and uncertainty into scoring functions to determine the best action. We also introduce an on-line help message generation method for error recovery which helps users operate the system and know the system's limitations quickly. The basic idea of our approach is, for novice users, to generate the feedbacks for system guidelines about what to say to achieve their goals at the current dialog state. In fact, this is an extended idea from Compute-Assisted Language Learning (CALL). A CALL system includes a tutor to give hints for students when they cannot continue the current dialog scenario. Similarly, when the user wants to access information of interest using our system, they can operate it easily by learning how to use the system via the help messages from the *AgendaHelp* and *UtterHelp* recovery strategies. These messages effectively help users because the current turn should generally be coherent to the previous turn to complete task.

8.2. Experiences

Experimental results show that supporting *n*-best recognition hypotheses increases the robustness to speech recognition errors, yielding shorter dialogs with a higher task completion rate. In addition, providing messages from the *UtterHelp* and *AgendaHelp* recovery strategies help users to follow the allowable dialog flows. The experimental results show that the use of an agenda graph as prior knowledge and help messages as error recovery strategies improves the performance of spoken dialog systems against noisy or unexpected inputs. We also anticipated that this effect would be more marked in earlier trials than in the later ones because users become skillful thorough the recommendation of possible subtasks and templates. In addition, using 5-best hypotheses was enough to cope with recognition errors when WER was about 20% in our dialog simulation. We think that a large number of hypotheses is not advisable due to the cost of complexity in time and resources. In addition, we employed about 900 examples in 200 dialogs to build the spoken dialog system of building guidance domain. We think that more examples can improve the dialog performance by covering more situations. However, there is a trade-off between the number of dialog examples and computational costs. If the dialog system holds too many examples, the number of candidate examples is increased, and more computational time is required to calculate their scores at the example selection step. In this paper, the size of about 900 examples yielded a good performance by covering various utterance patterns of simulated dialogs. We also performed an evaluation with real users. Real user evaluation showed similar results to the simulation test.

8.3. Future work

There are several possible subjects for further research on our approach. One of the further studies is to learn the optimal weights to obtain the best performance. The weights for the score functions defined in this study should be determined by as-yet undeveloped heuristics in developing task-oriented spoken dialog systems. In addition, a simple grid search was used to determine a set of weights to compute the node and example scores with the development set of simulated dialogs. It may offer some protection against local minima, but it is not very efficient. In fact, it is intractable to optimize the average scores over the weights on real numbers. Therefore, we have focused on learning the weights from dialog simulation by using an automatic and tractable method. In addition, manually building such graphs for various applications may be labor intensive and time consuming. We are addressing the problem of automatic knowledge acquisition of agenda graphs to structure task-oriented dialogs. Thus we are investigating how to build this graph automatically by clustering dialog states and utterances from annotated dialog corpus. Finally, future work will include expanding our system to other applications such as in-car navigation systems or weather information access systems.

Acknowledgement

This research was supported by the MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National Industry Promotion Agency) (NIPA-2009-C1090-0902-0045).

Appendix A. Sample conversation of building guidance domain

(1) **R**(obot): Hi. I am the PHOPE guidance robot in the Pohang Institute of Intelligent Robotics.
(KOR: 안녕하세요. 저는 포항 지능 로봇 연구소의 PHOPE 라는 안내 로봇입니다.)
(ROM: an-nyeongha-se-yo. jeo-neun po-hang ji-neung lo-bos yeon-gu-so-ui PHOPEla-neun an-nae lo-bos-ib-ni-da.)
[SA=GREET]
(2) **U**(ser): I want to reserve a meeting room for a group meeting.
(KOR: 그룹 미팅 예약하기 위한 회의실을 찾고 있어요.)
(ROM: geu-lub mi-ting ye-yag-ha-gi wi-han hoe-ui-sil-eul chajgo iss-eo-yo.)
[DA=STATEMENT][MG=SEARCH_LOC][ROOM_TYPE=meeting room]
(3) **R**: There are three meeting rooms: the Conference room, Meeting room A, and Meeting room B.
(KOR: 3개의 회의실이 존재합니다. 대회의실, 회의실 A, 회의실 B 입니다.)
(ROM: 3gae-ui hoe-ui-sil-i jon-jae-hab-ni-da. dae-hoe-ui-sil, hoe-ui-sil A, hoe-ui-sil B ib-ni-da.)
[SA=INFORM(ROOM_NAME)]
(4) **U**: How many persons can use the Conference room?
(KOR: 대회의실에는 몇 명이 들어갈 수 있나요?)
(ROM: dae-hoe-ui-sil-e-neun myeoch myeong-i deul-eo-gal su issna-yo?)
[DA=WH-QUESTION][MG=INFO_LOC][ROOM_NAME=Conference room]
(5) **R**: The Conference room can handle about 50 persons.
(KOR: 대회의실은 약 50 명의 사람들을 수용할 수 있습니다.)
(ROM: dae-hoe-ui-sil-eun yag 50myeong-ui sa-lamdeul-eul su-yonghal su iss-seub-ni-da.)
[SA=INFORM(ROOM_INFO)]
(6) **U**: Can I go there?
(KOR: 거기로 안내해주실 수 있나요?)
(ROM: geo-gi-lo an-nae-hae-ju-sil su issna-yo?)
[DA=YN-QUESTION][MG=GUIDE_LOC][LOC_REF=there]
(7) **R**: Yes. I will guide you to the Conference room. Follow me.
(KOR: 네. 대회의실로 안내해드리겠습니다. 저를 따라오세요.)
(ROM: ne. dae-hoe-ui-sil-lo an-nae-hae-deu-li-gess-seub-ni-da. jeo-leul tta-la-o-se-yo.)
[SA=GUIDE(ROOM)].

References

Bohus, B., Rudnicky, A., 2003. RavenClaw: dialog management using hierarchical task decomposition and an expectation agenda. In: Proc. of the European Conference on Speech, Communication and Technology, pp. 597–600.

Bohus, D., Rudnicky, A.I., 2005. Error handling in the RavenClaw dialog management framework. In: Proc. of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, pp. 225–232.

Fukubayashi, Y., Komatani, K., Ogata, T., Okuno, H., 2006. Dynamic help generation by estimating user's mental model in spoken dialogue systems. In: Proc. of the International Conference on Spoken Language Processing, pp. 1946–1949.

Griol, D., Hurtado, L.F., Segarra, E., Sanchis, E., 2008. A statistical approach to spoken dialog systems design and evaluation. Speech Communication 50 (8–9), 666–682.

Hazen, T.J., Seneff, S., Polifroni, J., 2002. Recognition confidence scoring and its use in speech understanding systems. Computer Speech and Language 16 (1), 49–67.

Heeman, P.A., 2007. Combining reinforcement learning with information-state update rules. In: Proc. of the Human Language Technology/North American Chapter of the Association for Computational Linguistics, pp. 268–275.

Hockey, B., Lemon, O., Campana, E., Hiatt, L., Hieronymus, J., Gruenstein, A., Dowding, J., 2003. Targeted help for spoken dialogue systems: intelligent feedback improves naive users' performance. In: Proc. of the European Chapter of Association of Computational Linguistics, pp. 147–154.

Jeong, M., Lee, G.G., 2008. Triangular-chain conditional random fields. IEEE Transactions on Audio, Speech and Language Processing 16 (7), 1287–1302.

Jung, S., Lee, C., Lee, G.G., 2008. Using utterance and semantic level confidence for interactive spoken dialog clarification. Journal of Computing Science and Engineering 2 (1), 1–25.

Jung, S., Lee, C., Kim, K., Jeong, M., Lee, G.G., 2009. Data-driven user simulation for automated evaluation of spoken dialog systems. Computer Speech and Language 23 (4), 479–509.

Kamm, C.A., Litman, D.J., Walker, M.A., 1998. From novice to expert: the effect of tutorials on user expertise with spoken dialogue systems. In: Proceedings of the International Conference on Spoken Language Processing, pp. 1211–1214.

Kim, B., 2002. Prosody/Phoneme Generation for Korean TTS (Text-to-Speech), Ph.D. Thesis, POSTECH.

Koo, M.W., Lee, C.H., Juang, B.H., 2001. Speech recognition and utterance verification based on a generalized confidence score. IEEE Transactions on Speech and Audio Processing 9 (8), 821–832.

Lee, K.-N., Chung, M., 2007. Morpheme-based modeling of pronunciation variation for large vocabulary continuous speech recognition in Korean. IEICE Transactions on Information and System E90-D (7), 1063–1072.

Lee, C., Jung, S., Lee, D., Lee, G.G., 2007. Example-based error recovery strategy for spoken dialog system. In: Proc. of the IEEE Workshop on Automatic Speech Recognition and Understanding, pp. 538–543.

Lee, C., Jung, S., Kim, S., Lee, G.G., 2009. Example-based dialog modeling for practical multi-domain dialog system. Speech Communication 51 (5), 466–484.

Lesh, N., Rich, C., Sidner, C., 2001. Collaborating with focused and unfocused users under imperfect communication. In: Proceedings of the International Conference on User Modeling, pp. 63–74.

Levin, E., Pieraccinin, R., Eckert, W., 2000. A stochastic model of computer–human interaction for learning dialog strategies. IEEE Transactions on Speech and Audio Processing 8 (1), 11–23.

Lo, W., Soong, F., 2005. Generalized posterior probability for minimum error verification of recognized sentences. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 85–88.

Lochbaum, K.E., 1998. A collaborative planning model of intentional structure. Computational Linguistics 24 (4), 525–572.

Mctear, M., 1998. Modelling spoken dialogues with state transition diagrams: experience of the CSLU toolkit. In: Proceedings of the International Conference on Spoken Language Processing, pp. 1223–1226.

Mctear, M., O'neill, I., Hanna, P., Liu, X., 2005. Handling errors and determining confirmation strategies – an object-based approach. Speech Communication 45 (3), 249–269.

Nagao, M., 1984. A frame work of a mechanical translation between Japanese and English by analogy principle. In: Proceedings of the International NATO Symposium on Artificial and Human Intelligence, pp. 173–180.

Oh, S., Choi, Y.-H., Yun, S., Jun, B.-J., Lee, C., Jang, H., Song, J., Kang, J.-G., Choi, W.-S., An, S.-Y., 2009. A system architecture for intelligent building guide robot PHOPE. In: Proceedings of the International Conference on Autonomous Robots and Agents.

Paek, T., 2006. Reinforcement learning for spoken dialogue systems: comparing strengths and weaknesses for practical deployment. In: Proceedings of Workshop on Dialogue on Dialogues, International Conference of Spoken Language Processing.

Rich, C., Sidner, C., 1998. Collagen: a collaboration agent for software interface agents. Journal of User Modeling and User-Adapted Interaction 8 (3), 315–350.

Roy, S., Subramaniam, L.V., 2006. Automatic generation of domain models for call centers from noisy transcriptions. In: Proceeding of the Association for Computational Linguistics, pp. 737–744.

Schatzmann, J., Thomson, B., Weilhammer, K., Ye, H., Young, S., 2007. Agenda-based user simulation for bootstrapping a POMDP dialogue system. In: Proceedings of the Human Language Technology/North American Chapter of the Association for Computational Linguistics, pp. 149–152.

Shin, J., Narayanan, S., Gerber, L., Kazemzadeh, A., Byrd, D., 2002. Analysis of user behavior under error conditions in spoken dialogs. In: Proceedings of the International Conference on Spoken Language Processing, pp. 2069–2072.

Skantze, G., 2005. Exploring human error recovery strategies: implications for spoken dialogue systems. Speech Communication 45 (3), 325–341.

Torres, F., Hurtado, L., Garcia, F., Sanchis, E., Segarra, E., 2005. Error handling in a stochastic dialog system through confidence measures. Speech Communication 45 (3), 211–229.

Walker, M., Langkilde, I., Wright, J., Gorin, A., Litman, D., 2000. Learning to predict problematic situations in a spoken dialogue system: experiments with how may I help you? In: Proceedings of the North American Chapter of the Association for Computational Linguistics, pp. 210–217.

Williams, J.D., 2008. The best of both worlds: Unifying conventional dialog systems and POMDPs. In: Proceedings of the International Conference on Spoken Language Processing.

Williams, J.D., Young, S., 2007. Partially observable Markov decision processes for spoken dialog systems. Computer Speech and Language 21, 393–422.

Young, S., Schatzmann, J., Weilhammer, K., Ye, H., 2007. The hidden information state approach to dialog management. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 149–152.