Available online at www.sciencedirect.com

ScienceDirect

Speech Communication 51 (2009) 466–484

SPEECH COMMUNICATION

www.elsevier.com/locate/specom

Example-based dialog modeling for practical multi-domain dialog system

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

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

Received 24 April 2008; received in revised form 28 December 2008; accepted 23 January 2009

Abstract

This paper proposes a generic dialog modeling framework for a multi-domain dialog system to simultaneously manage goal-oriented and chat dialogs for both information access and entertainment. We developed a dialog modeling technique using an example-based approach to implement multiple applications such as car navigation, weather information, TV program guidance, and chatbot. Example-based dialog modeling (EBDM) is a simple and effective method for prototyping and deploying of various dialog systems. This paper also introduces the system architecture of multi-domain dialog systems using the EBDM framework and the domain spotting technique. In our experiments, we evaluate our system using both simulated and real users. We expect that our approach can support flexible management of multi-domain dialogs on the same framework.
© 2009 Elsevier B.V. All rights reserved.

Keywords: Example-based dialog modeling; Generic dialog modeling; Multi-domain dialog system; Domain identification

1. Introduction

Use of natural language[1] in human–computer interaction is attractive because it is one of the most natural, efficient, and flexible means for people to communicate with each other. The objective in developing natural language dialog systems is to provide a natural way for any user to access and manage information. These systems are becoming ubiquitous due to their rapid improvement in performance and decrease in cost.

The performance of natural language dialog systems has improved over time and the applications of these systems have become more general. Early natural language dialog systems functioned in restricted domains such as telephone-based weather information systems (JUPITER) (Zue et al., 2000) and travel planning (DARAPA communicator) (Walker et al., 2001). More recently developed systems are used in in-car navigation, entertainment, and communications (Minker et al., 2004; Lemon et al., 2006; Weng et al., 2006). For example, the EU project TALK[2] focused on the development of new technologies for adaptive dialogue systems using speech, graphics, or a combination of the two in the car. Chatbots (e.g., ALICE[3]) have been also developed for application to entertainment and education, and as web agents.

More recently, multi-domain dialog systems have been employed in real life situations (Allen et al., 2000; Larsson and Ericsson, 2002; Lemon et al., 2002; Pakucs, 2003; Komatani et al., 2006). Such multi-domain dialog systems are now able to provide services for telematics, smart home, or intelligent robots. These systems have gradually become capable of supporting multiple tasks and of accessing information from a broad variety of sources and

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

[1] In this paper, we will use the term *natural language* to include both *spoken language* and *written language*.

[2] TALK project site: http://www.talk-project.org.

[3] ALICE chatbot site: http://www.alicebot.org/.

0167-6393/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.specom.2009.01.008

services. However, building a dialog management system for the processing of dynamic multi-domain dialogs is difficult. One difficulty is identifying the user's domain of interest correctly and switching between domains smoothly. Solving this problem is critical to improve the usability of multi-domain dialog systems. In addition, dialog systems must be made capable of adapting and porting to new domains. A need has developed for a generic dialog modeling that can be used for rapid and efficient development of dialog managers.

In this paper, we address the challenge of supporting domain portable modeling and developing chatbot entertainment systems. Dialogs were conducted in Korean. We designed and investigated an example-based approach as a generic dialog model. By this approach, we can handle different dialog genres and domains including both chat and goal-oriented dialogs. Example-based dialog modeling (EBDM) is one of several data-driven methods for deploying dialog systems. The basic idea of our approach is that a dialog manager (DM) uses dialog examples that are semantically indexed to a database, instead of domain-specific rules or probabilistic models for dialog management. We have presented the EBDM methodology for a goal-oriented dialog system in a single electronic program guide (EPG) domain (Lee et al., 2006). In this paper, we extend our previous work to explore a generic dialog modeling framework for managing multi-domain goal-oriented dialogs and chat dialogs in the same framework, and we adapt rephrasing strategy to handle recognition or understanding errors in the DM because this strategy is a common response to repair errors in spoken dialog systems (Shin et al., 2002). We have developed several applications in different domains for case studies (e.g., car navigation, weather information, etc.) using the EBDM framework. To facilitate multi-domain dialog systems, we also developed a classification module, called a spotter module, that determines the target agent or domain class of the user input. The spotter modules use a combination of keyword spotting and feature-based classification to allow the DM to identify an appropriate dialog expert that can be used to manage the current input.

Evaluating dialog systems is very difficult because of high cost and the need for objectivity. One method to evaluate the system is to use a human judge. Evaluation criteria include the task completion rate (TCR), total elapsed time and the opinions of judges as assessed by questionnaire. However, when humans are used to evaluate dialog systemsonces, objectivity become questionable; the process is also expensive and time-consuming. An alternative approach is to evaluate automatic system-to-system dialog by simulated users (Watanabe et al., 1998; Lopez-Cozar et al., 2003; Schatzmann et al., 2005; Georgila et al., 2005). This is a faster and less costly evaluation method than using human judges. In this study, we evaluated our system using both real and simulated users.

This paper is organized as follows. Previous related work is described in Section 2, followed by the methods and advantages of the EBDM in Section 3. Case studies of using our approach are presented in Section 4. We introduce our system architecture for multi-domain dialog systems in Section 5. Section 6 explains about corpus collection and system implementation to build dialog systems. Section 7 provides the results of a simulated user and real user evaluation to verify our approach. Finally, we draw conclusions and make suggestions for future work in Section 8.

2. Related work

Early dialog systems such as SUNDIAL (Peckham, 1993) and ARISE (Lamel et al., 1999) were designed by application developers who have domain-specific knowledge. These systems are usually confined to highly structured tasks, where a restricted and regularized language set can be expected. This knowledge-based approach generally uses finite-state automata which often involve handcrafted rules. These are dictated by the knowledge of the application, and by continuous experiments with real users. It has been used for rapid prototyping of dialog systems for strong-typed interactions with clearly-defined structures and goals (McTear, 1998). This approach has also been deployed in many practical applications. However, this method has the problem that hand-crafting rules in advance is difficult. It also suffers from poor domain portability: when the designers develop a new application for a different domain, the entire design process must be restarted from the beginning.

Traditional knowledge-based approaches also do not avoid this problem of poor domain portability. To overcome this limitation, several groups (Rich and Sidner, 1998; Bohus and Rudnicky, 2003; Bui et al., 2004; Larsson and Traum, 2006) have explored generic dialog modeling approaches based on agendas or task models, which are powerful representations for segmenting large tasks into smaller and more easily handled subtasks. Several extensions are being investigated by using this approach. However, the design process is still time-consuming and expensive because the knowledge sources (e.g., hierarchical task structure and plan recipes) are usually designed by human experts. However, our approach can reduce the cost of rebuilding and maintaining the dialog systems because the dialog strategies are automatically determined by a small number of dialog examples.

More recently, the research community for dialog management has exploited the benefits of data-driven approaches to automatic speech recognition (ASR) and spoken language understanding (SLU). Although a data-driven approach requires time-consuming data annotation, the training is done automatically and requires little human supervision. In addition, new systems can be developed at the only cost of collecting new data for moving to a new domain; this requires less time and effort than the knowledge-based approach. These advantages have motivated the development of stochastic dialog modeling using reinforcement learning (RL) based on Markov decision processes (MDPs) or partially observable MDPs (POMDPs) (Levin et al., 2000; Williams and Young, 2007). These frameworks apply statistically data-driven and theoretically-principled dialog modeling to dynamically allow changes to the dialog strategy. They accomplish this by optimizing some reward or cost functions given the current dialog state. However, practical deployment of RL in dialog systems has encountered several obstacles (Paek, 2006). For example, the optimized policy may remove control from application developers. Refining the dialog control is also difficult. These are serious problems because the developers should have the opportunities to easily control the dialog flows in practical systems. Although many researchers are solving these problems in ongoing work (Williams and Young, 2005; Young et al., 2007; Thomson et al., 2008), this approach still needs improvement before it can be applied to develop practical dialog systems. On the other hand, in our approach, a system developer can flexibly determine a set of state variables without a complexity problem, and the dialog flows can be easily controlled with the dialog examples. In addition, traditional RL-based dialog systems require a large number of dialog corpora to learn an optimal policy because of a very large state space and a very large policy space. To address this problem, a hybrid approach to integrate reinforcement and supervised learning has been investigated to optimize dialog policies with a fixed dialog corpus (Henderson et al., 2005). This approach can eliminate the need for a large number of dialog corpora to optimize the dialog policies in traditional RL-based dialog systems. In this approach, RL is used to optimize a measure of dialog reward, while supervised learning is used to restrict the learnt policy to the portion of the space for which data are available. Our approach also allows development of goal-oriented dialog systems using a fixed human–human dialog corpus (<150 dialogs in our experiments) by using a semantic-based indexing scheme (Section 3.2).

A new supervised approach to dialog management has been developed which uses maximum likelihood estimation of a stochastic model from human–human dialog corpus (Hurtado et al., 2005). To avoid the data sparseness problem, this approach uses dialog register (DR) representation, which is a data structure for keeping track of discourse history as dialog state sequences. The DR contains the information about slot names and slot values provided by the user throughout the previous history of the dialog. Our approach is similar to that proposed by Hurtado et al. (2005) because we also use a relatively small number of human–human dialog corpora and discourse history vector (DHV) representation to take into account the information supplied by the evolution of the dialog (Section 3.2). However, any fixed dataset will only provide information about a small portion of the huge dialog state space. To address this problem, our approach can directly verify whether the current input is correct by computing text-to-text similarity. If the DM determines the current input is out-of-coverage pattern, then the DM recommends an in-coverage template to continue the current dialog given the dialog state (Section 3.3.3). Moreover, according to the characteristics of the dialog's task, our method allows to easily integrate heuristics into query generation and similarity measures when the DM finds relevant dialog examples (Sections 3.3.1 and 3.3.2).

Other example-based approaches exist that perform dialog modeling using dialog examples (Inui et al., 2001; Murao et al., 2003; Jenkins et al., 2007). For example, service-oriented chatbot systems have been developed to help users access information from a web site more easily (Jenkins et al., 2007). In these systems, keyword spotting techniques are commonly used to generate a system response by finding template, and the matching is performed by applying heuristic pattern-matching rules to the user's input. These approaches attempt to select the best template using keywords and dialog acts. We have improved these approaches to support multiple applications of natural language dialog systems by incorporating several discourse features and heuristic strategies.

3. Example-based dialog modeling

3.1. Motivation

This section introduces our generic dialog modeling framework for multi-domain dialog systems. We developed a situation-based dialog management to lead dialog flows using situation-based rules. In our system, the situation is defined as the dialog state space which determines the next system action. This state space is represented by several state variables: user utterance, user intention, a set of semantic slots and values, discourse history, the previous system action, and the results of queries to the domain knowledge database. This situation-based dialog management leads the dialog using a set of condition–action rules. We implemented an object-oriented architecture for easy domain extension and portability (O'Neil et al., 2005) (Fig. 1). However, because of the poor adaptability of knowledge-based approaches to multi-domain dialog systems, the number of situation-based rules would have to increase. Furthermore, designing more flexible and natural dialog flows using the different sets of dialog state variables is very difficult. To avoid these limitations, we have proposed EBDM for automatically predicting the next system actions. EBDM 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 from a large parallel corpus, without knowledge of the language's structure. The idea of EBMT can be extended to determine subsequent system actions by finding similar dialog examples within the dialog corpus. The system actions can be predicted by finding semantically similar user utterances in a set of state variables.

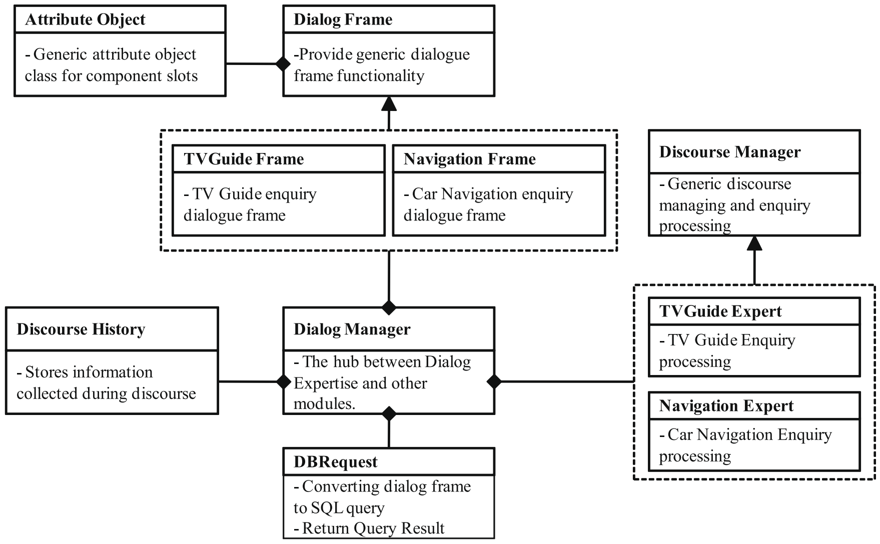
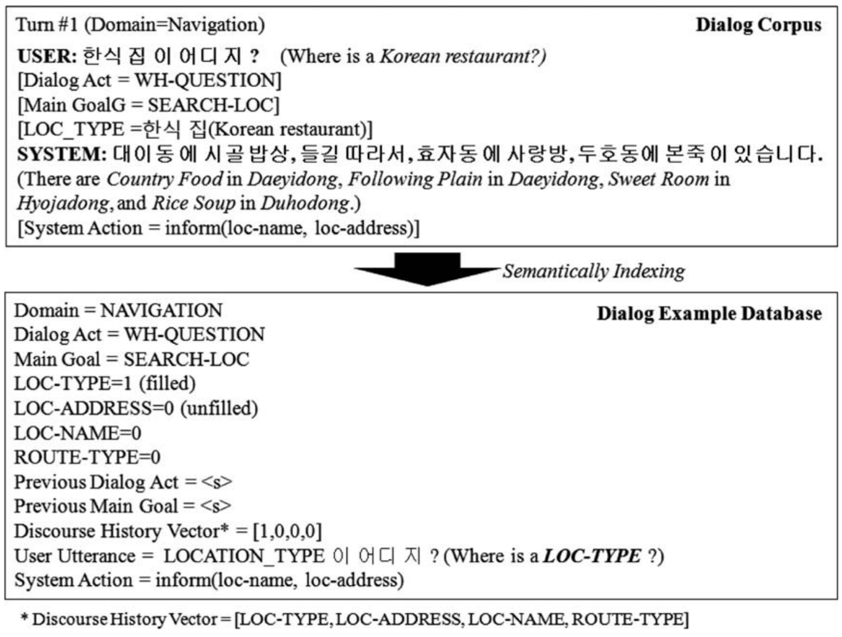

Attribute Object
- Generic attribute object class for component slots
Dialog Frame
-Provide generic dialogue frame functionality
TVGuide Frame
- TV Guide enquiry dialogue frame
Navigation Frame
- Car Navigation enquiry dialogue frame
Discourse Manager
- Generic discourse managing and enquiry processing
Discourse History
- Stores information collected during discourse
Dialog Manager
- The hub between Dialog Expertise and other modules.
TVGuide Expert
- TV Guide Enquiry processing
Navigation Expert
- Car Navigation Enquiry processing
DBRequest
- Converting dialog frame to SQL query
- Return Query Result

Fig. 1. Object-oriented architecture of situation-based dialog system for EPG service and car navigation domains.

3.2. Building the dialog example database

First we built a dialog example database (DEDB) by collecting human–human dialog corpora related to predefined scenarios. For the experiments reported in this paper, we used about 120 dialogs for each goal-oriented domain. Then we manually assigned semantic tags (e.g., dialog act, main goal, and slot entities) to the user utterances, and system action tags to the system utterances (top of Fig. 2). We also used a hand-crafted automatic system to extract semantic and discourse features (e.g., previous intention and slot filling status); these features are helpful in determining the next system actions. We can automatically extract these features by keeping track of the dialog states for each point in the dialog (bottom of Fig. 2). Then the DEDB is semantically indexed to generalize the data in which index constraints can be determined according to state variables chosen by a system designer for domain-specific applications. Each turn pair (user turn, system turn) in the dialog corpora is mapped to semantic instances in the DEDB (Fig. 2). The index constraints represent the state variables, which are domain-independent attributes. Our basic constraints consist of general features to define the dialog states such as domain, the current user intention (dialog act and main goal), slot flags of the current utterance, discourse history vector, and the lexico-semantic string of the utterance (Table 1).


Turn #1 (Domain=Navigation) **Dialog Corpus**
USER: 한식 집 이 어디 지 ? (Where is a *Korean restaurant?)*
[Dialog Act = WH-QUESTION]
[Main GoalG = SEARCH-LOC]
[LOC_TYPE =한식 집(Korean restaurant)]
SYSTEM: 대이동 에 시골밥상, 들길 따라서, 효자동 에 사랑방, 두호동에 본죽 이 있습니다.
(There are *Country Food* in *Daeyidong*, *Following Plain* in *Daeyidong*, *Sweet Room* in *Hyojadong*, and *Rice Soup* in *Duhodong*.)
[System Action = inform(loc-name, loc-address)]

Semantically Indexing

Domain = NAVIGATION **Dialog Example Database**
Dialog Act = WH-QUESTION
Main Goal = SEARCH-LOC
LOC-TYPE=1 (filled)
LOC-ADDRESS=0 (unfilled)
LOC-NAME=0
ROUTE-TYPE=0
Previous Dialog Act = <s>
Previous Main Goal = <s>
Discourse History Vector* = [1,0,0,0]
User Utterance = LOCATION_TYPE 이 어디 지 ? (Where is a ***LOC-TYPE*** ?)
System Action = inform(loc-name, loc-address)

* Discourse History Vector = [LOC-TYPE, LOC-ADDRESS, LOC-NAME, ROUTE-TYPE]

Fig. 2. Indexing scheme for dialog example database on car navigation domain.

Table 1
Index and search constraints for example-based dialog modeling.

Constraints	Detail descriptions
Agent (AG)	Dialog's genre of the current utterance necessary to adapt to multi-domain dialog system (e.g., *chat* and *task*)
Domain (DO)	Detailed topic of the current utterance necessary to adapt to multi-domain dialog system (e.g., *TV*, *navigation*, *weather*, *sports*, etc.)
User utterance	Lexico-semantic utterance in which the values of domain-specific component slots on user utterance are replaced by a slot name (e.g., "Korean restaurant" is replaced by *LOC-TYPE*)
Dialog act (DA)	Domain-independent label of an utterance at the level of illocutionary force (e.g., STATEMENT, REQUEST, WH-QUESTION, etc.)
Main goal (MG)	Domain-specific user goal of an utterance (e.g., GUIDE-LOC, SEARCH-LOC, SEARCH-PHONE, etc.)
Slot flag (SF)	Filling flag of the corresponding slot name (e.g., If user says "Korean restaurant" in the current utterance, the flag of *LOC-TYPE* is set to 1)
Previous dialog act (PREV_DA)	Dialog act of the previous user utterance
Previous main goal (PREV_MG)	Main goal of the previous user utterance
Discourse history vector (DHV)	Binary vector for slot-filling status that is assigned a value of 1 if the component slot is already filled, and 0 otherwise

3.3. Dialog management processes

To determine the next system actions in the dialog manager, the EBDM framework uses four processes as follows:

- *Query generation*: The dialog manager generates a structured query language (SQL) statement using discourse history and the current dialog frame.
- *Example search*: The dialog manager searches for semantically similar dialog examples in the DEDB given the current dialog state. If no examples are retrieved, some state variables can be removed by relaxing particular variables according to the level of importance given the dialog's genre and domain.
- *Example selection*: The dialog manager selects the best example to maximize the utterance similarity measure based on lexico-semantic similarity and discourse history similarity.
- *Example recommendation*: When no examples are found, the dialog manager tries to recommend the most relevant template of dialog examples which can be processed at the current turn.

The overall strategy of the EBDM framework (Fig. 3) can be divided into several processes. We will explain the details of each process in the following sections.

3.3.1. Query generation and example search

The index keys are also used to query the DEDB to search for dialog examples that are similar to the current dialog state. The query keys are extracted from SLU results (e.g., user intention and semantic frame), domain, and discourse history (e.g., previous user intention and discourse history vector). For example, if a user in a car navigation domain says "*Let me go to Country Food in Daeyidong*", the DEDB retrieves a candidate list of examples that are related to the current dialog state (Fig. 4). Two kinds of matches can occur: (1) exact match and (2) partial match.

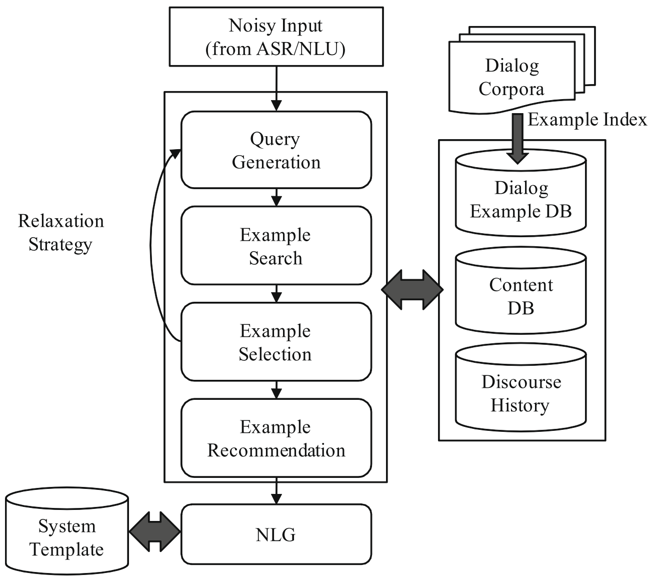
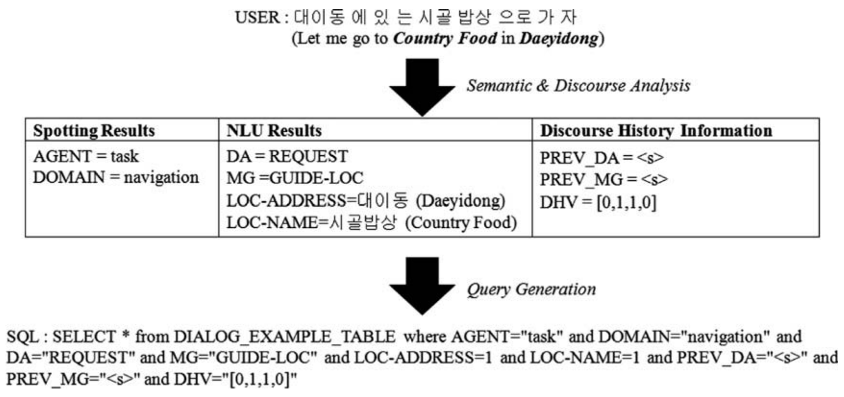


Fig. 3. Strategy of the example-based dialog modeling (EBDM) framework.

During the example search, EBDM first scans the DEDB for utterances that match all query constraints exactly. If it does not find an exact match, it applies a relaxation strategy to remove particular query constraints, then searches for similar utterances. The aim of each relaxation strategy is to search for more dialog examples by excluding some constraints. The constraints to be eliminated are selected by their importance and reliability according to the genre and domain of the dialog.

In general, all constraints in the different domains are not equally important to determine the next system action, because the dialog of each domain has its own characteristics. For example, a slot-filling task (e.g., car navigation and flight reservation) shows transactional dialog flow because this task usually has preference orders to fill slots. In this case, because the DHV constraint is more important than any other constraints, it is retained when searching for


USER : 대이동 에 있 는 시골 밥상 으로 가 자
(Let me go to ***Country Food*** in ***Daeyidong***)

Spotting Results	**NLU Results**	**Discourse History Information**

AGENT = task
DOMAIN = navigation
	DA = REQUEST
	MG =GUIDE-LOC
	LOC-ADDRESS=대이동 (Daeyidong)
	LOC-NAME=시골밥상 (Country Food)
		PREV_DA = <s>
		PREV_MG = <s>
		DHV = [0,1,1,0]

SQL : SELECT * from DIALOG_EXAMPLE_TABLE where AGENT="task" and DOMAIN="navigation" and DA="REQUEST" and MG="GUIDE-LOC" and LOC-ADDRESS=1 and LOC-NAME=1 and PREV_DA="<s>" and PREV_MG="<s>" and DHV="[0,1,1,0]"

Fig. 4. Example of query generation on car navigation domain.

partial match in the slot-filling task. In addition, among a set of constraints, the more accurate ones are considered more reliable. System developers can identify the reliable constraints according to the performance of each module. For example, the performance of dialog act recognition problem may be significantly higher than that of a named entity (e.g., domain-specific slot entity) recognition problem (Eun et al., 2005). In this case, we first relaxed the slot flag (SF) constraints because errors can occur as a result of the SLU module.

Although an exact match often guarantees more precise prediction of the next system action when all constraints are correctly extracted by SLU results and discourse history, the relaxation strategy is necessary for the following reasons:

- *Robustness*: The query constraints may not be perfectly reliable. The query constraints can contain errors which are caused by a noisy environment or unexpected inputs through ASR and SLU modules. Even if the prior modules give incorrect results, the EBDM framework tries to find similar examples by exact match. The exact match may return results which contain dialog examples which are irrelevant to the current state. These examples are highly likely to be rejected later by example selection because the utterance similarity is lower than a threshold (Section 3.3.2). In this case, the system removes some constraints that are less reliable and less important, then searches a different candidate set of dialog examples to find a partial match.
- *Data sparseness*: Preparing a large number of dialog corpora is difficult because corpus collection and annotation are time-consuming and labor-intensive tasks. Furthermore, covering all possible situations for dialogs with a fixed dialog corpus is not possible. Thus, the relaxation strategy must be used to solve the problem of data sparseness. In the relaxation step, a set of constraints for the exact match can be approximated by a set of more reliable and important constraints to determine the next system action. The relaxed constraints can return more results by covering a larger state space. A relaxed set of constraints may involve the current dialog act and main goal because the system actions mainly depend on user intention at the current turn, and because these constraints are commonly more reliable than any other constraints.

The relaxation strategy differs according to the dialog domain and genre of a given application. We will describe the relaxation strategies of different applications in our case studies (Section 4).

3.3.2. Example selection

After searching the candidate examples, we use the example score to select the best dialog example e^* among a set of the candidate examples E

$$e^* = \underset{e_i \in E}{\arg\max}\, S_{utter}(u, e_i) \tag{1}$$

$$= \alpha S_{LSS}(w_u, w_{e_i}) + (1-\alpha) S_{DHS}(v_u, v_{e_i}) \tag{2}$$

where w_u is the current user utterance input to the system from a keyboard or through a speech recognizer and w_{e_i} is the user utterance of a dialog example in E. $S_{utter}(u, e_i)$ denotes the value of the utterance similarity of the user utterance u and dialog example e_i. The utterance similarity measure includes the lexico-semantic similarity S_{LSS} and the discourse history similarity S_{DHS} (Fig. 5).

The lexico-semantic similarity $S_{LSS}(w_u, w_e)$ is defined as a normalized edit distance between lexico-semantic utterance of the current user's utterance (w_u), and the example utterance (w_e). In normalized edit distance, a cost function C is defined as:

$$C(i,j) = \begin{cases} 0 & \text{if } w_{u,i} = w_{e,j} \quad \text{and} \quad t_{u,i} = t_{e,j} \\ 0.5 & \text{if either } w_{u,i} = w_{e,j} \quad \text{or} \quad t_{u,i} = t_{e,j} \\ 1 & \text{if } w_{u,i} \neq w_{e,j} \quad \text{and} \quad t_{u,i} \neq t_{e,j} \end{cases} \tag{3}$$

where $w_{u,i}$ denotes the morpheme surface of the user's utterance at position i and $w_{e,j}$ denotes the morpheme surface of the example utterance at position j, $t_{u,i}$ represents the POS tag of the user's utterance at position i, and $t_{e,j}$ represents the POS tag of the example utterance at position

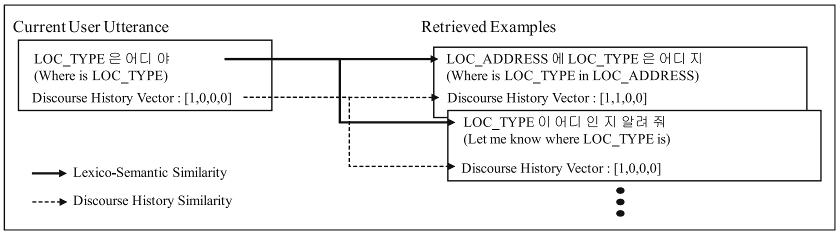

Current User Utterance
Retrieved Examples
LOC_TYPE 은 어디 야
(Where is LOC_TYPE)
Discourse History Vector : [1,0,0,0]
LOC_ADDRESS 에 LOC_TYPE 은 어디 지
(Where is LOC_TYPE in LOC_ADDRESS)
Discourse History Vector : [1,1,0,0]
LOC_TYPE 이 어디 인 지 알려 줘
(Let me know where LOC_TYPE is)
Discourse History Vector : [1,0,0,0]
Lexico-Semantic Similarity
Discourse History Similarity

Fig. 5. Utterance Similarity for example selection.

j. The cost function is defined as a morpheme and the morpheme's POS tag at the positions compared. We assign a higher cost when the morpheme and the POS tag differ in both the user utterance and the example utterance. However, the case when the same POS tag represents different morpheme surfaces gives smaller cost to improve the coverage by generalizing the morpheme surfaces. For example, case particles in Korean have different morpheme surfaces according to the final consonant of previous syllable. This similarity indicates that a dialog example is lexically closer to the current utterance. The lexico-semantic similarity is important to detect incorrect utterances containing ASR errors (e.g., deletions, insertions, and substitutions).

In calculating the utterance similarity, we consider an additional measure to reflect discourse information. We define the degree of the discourse history vector similarity $S_{DHS}(v_u, v_e)$ as follows:

$$S_{DHS}(v_u, v_e) = \frac{v_u \cdot v_e}{\|v_u\|\|v_e\|} \tag{4}$$

$$= \frac{\#\text{ of slot-filling states in common}}{\#\text{ of component slots}} \tag{5}$$

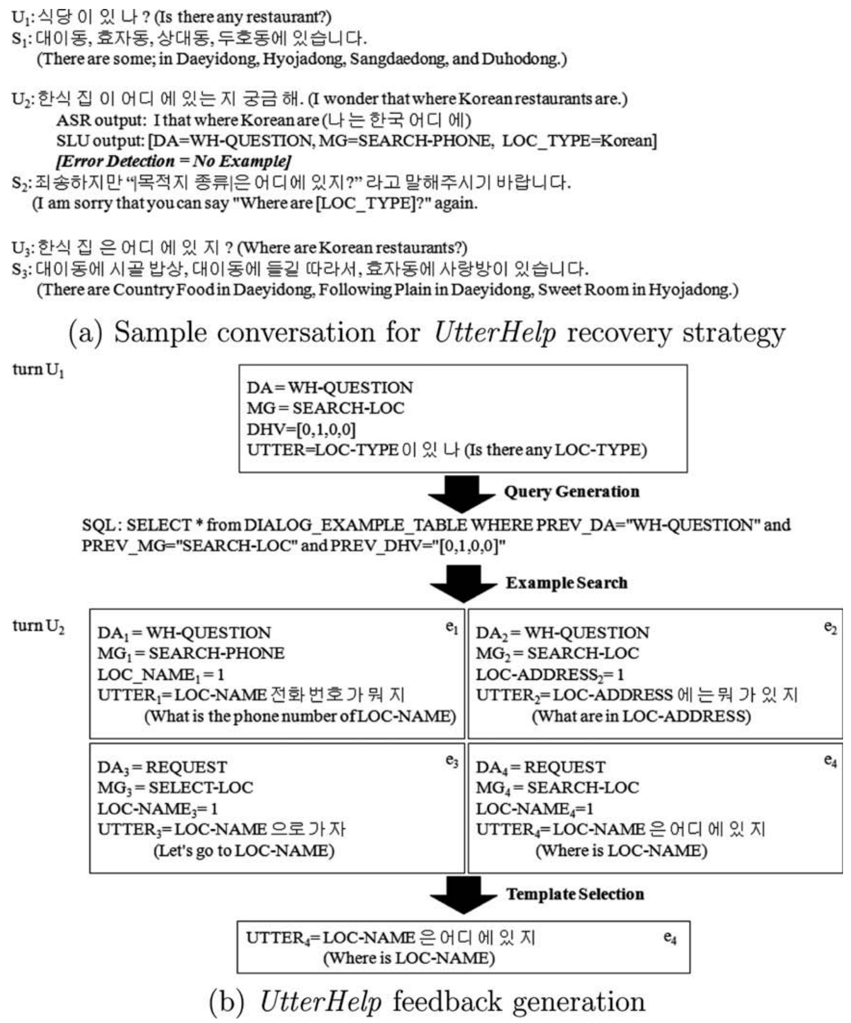
where v_u is the discourse history vector of the current dialog state, v_e is the discourse history vector of the example's dialog state, and $\|\cdot\|$ denotes a Euclidean norm. Eq. (4) describes a cosine measure between v_u and v_e and is a widely used measure of vector representation. These vectors are assigned a value of 1 if the component slot is already filled, and 0 otherwise. We use this measure to compare the discourse history of the current dialog state and the example's dialog state. Higher values of $S_{DHS}(v_u, v_e)$ means that the dialog example and the current dialog state are more similar.

Given two similarity measures, the utterance similarity can be expanded using linear interpolation of Eq. (1) with properly defined weights for each application. The value of α can be assigned based on empirical analysis of simulated human–computer dialogs. Alternatively, it can be assigned manually at the discretion of the system developer according to the characteristics of the dialog genre and task. For example, α can be set lower to manage transactional dialogs (e.g., a car navigation domain) in which the user utterance is highly correlated to the previous system utterance because this kind of task usually has preference orders to fill slots. After processing the example selection, the best example is used to generate the system actions. Then the template-based natural language generation module can generate system utterances by using system actions and retrieved contents.

3.3.3. Example recommendation

In our system, we assume that the user behaves optimally and consistently because typical users stay focused most of the time during imperfect communication when using natural language dialog systems (Lesh et al., 2001). However, some users, particularly novice users, can produce unexpected inputs which can induce errors in the ASR and SLU modules. This is a serious problem in the EBDM framework because it is impossible to cover all possible patterns using a fixed number of examples. In our framework, we define a *No Example* error when no dialog examples are retrieved after searches for exact and partial matches. *No Example* means that the system cannot find similar examples to determine the next system action. In general, the cause of wrong system responses may lie in out-of-coverage problems such as out-of-vocabulary (OOV) and out-of-grammar (OOG) in the recognition and understanding models. Non-expert users of dialogue systems may produce unexpected out-of-coverage inputs; when they do, recognition and understanding errors occur frequently. In the example selection step, these inputs should be rejected because the utterance similarity of candidate examples falls below a threshold, and the DM tries to search for more examples using the relaxation strategy. Nevertheless, if the results of a partial match are rejected or if there are no partial matches, we regard this situation as having potential errors because the utterance may be semantically or grammatically incorrect. This is similar to OOG in grammar-based recognition. In our framework, *No Example* errors can be easily detected by scoring the dialog examples using the similarity measure and we do not need extra knowledge sources such as grammar and domain concept representation. This error detection method is smoothly integrated into the EBDM framework by using the DEDB. For the *No Example* situation, we define an it UtterHelp error recovery strategy in which the DM gives an in-coverage example of what the user could say at the current dialog state. The dialog manager provides the in-coverage utterance template in the DEDB because the examples were used to train the ASR, SLU, and DM modules.

Fig. 6 illustrates the sample conversation and generation of the *UtterHelp* recovery strategy. The system responds with appropriate answers when the user utterance can be


U$_1$: 식당 이 있 나 ? (Is there any restaurant?)
S$_1$: 대이동, 효자동, 상대동, 두호동에 있습니다.
(There are some; in Daeyidong, Hyojadong, Sangdaedong, and Duhodong.)

U$_2$: 한식 집 이 어디 에 있는 지 궁금 해. (I wonder that where Korean restaurants are.)
ASR output: I that where Korean are (나 는 한국 어디 에)
SLU output: [DA=WH-QUESTION, MG=SEARCH-PHONE, LOC_TYPE=Korean]
[Error Detection = No Example]
S$_2$: 죄송하지만 "[목적지 종류]은 어디에 있지?" 라고 말해주시기 바랍니다.
(I am sorry that you can say "Where are [LOC_TYPE]?" again.

U$_3$: 한식 집 은 어디 에 있 지 ? (Where are Korean restaurants?)
S$_3$: 대이동에 시골 밥상, 대이동에 들길 따라서, 효자동에 사랑방이 있습니다.
(There are Country Food in Daeyidong, Following Plain in Daeyidong, Sweet Room in Hyojadong.)

(a) Sample conversation for *UtterHelp* recovery strategy

turn U$_1$

DA = WH-QUESTION
MG = SEARCH-LOC
DHV=[0,1,0,0]
UTTER=LOC-TYPE 이 있 나 (Is there any LOC-TYPE)

SQL: SELECT * from DIALOG_EXAMPLE_TABLE WHERE PREV_DA="WH-QUESTION" and PREV_MG="SEARCH-LOC" and PREV_DHV="[0,1,0,0]"

turn U$_2$

e$_1$
DA$_1$ = WH-QUESTION
MG$_1$ = SEARCH-PHONE
LOC_NAME$_1$ = 1
UTTER$_1$=LOC-NAME 전화 번호 가 뭐 지
(What is the phone number of LOC-NAME)

e$_2$
DA$_2$ = WH-QUESTION
MG$_2$ = SEARCH-LOC
LOC-ADDRESS$_2$= 1
UTTER$_2$=LOC-ADDRESS 에 는 뭐 가 있 지
(What are in LOC-ADDRESS)

e$_3$
DA$_3$ = REQUEST
MG$_3$ = SELECT-LOC
LOC-NAME$_3$= 1
UTTER$_3$=LOC-NAME 으로 가 자
(Let's go to LOC-NAME)

e$_4$
DA$_4$ = REQUEST
MG$_4$ = SEARCH-LOC
LOC-NAME$_4$=1
UTTER$_4$=LOC-NAME 은 어디 에 있 지
(Where is LOC-NAME)

e$_4$
UTTER$_4$=LOC-NAME 은 어디 에 있 지
(Where is LOC-NAME)

(b) *UtterHelp* feedback generation

Fig. 6. *UtterHelp* recovery strategy. U_i indicates a user turn at time i and S indicates a system turn at time i.

successfully processed by the natural language dialog system (turn U_1, Fig. 6a). But when errors occur in the recognition and understanding module (turn U_2, Fig. 6a), the system cannot find examples that have a similarity score that exceeds the threshold of utterance similarity, and a *No Example* error occurs. In this conversation, the main goal is not SEARCH-PHONE but SEARCH-LOC. The DM tries to find possible templates and to generate a help message prompt with an utterance template to help achieve the user's goal at turn S_2. This template is an in-coverage example which were used to train ASR, SLU, and DM modules. Then, the users can rephrase using the utterance template at turn U_3.

The template selection for *UtterHelp* recovery strategy is implemented as follows (Fig. 6b). We first consider the discourse context to select in-coverage examples from the DEDB based on the previous user turn. Because each example includes previous user intention (e.g., dialog act and main goal) and previous discourse history vector as query keys, the system can search possible dialog state spaces using the previous keys in the discourse history. Then, the possible dialog example candidates (e_1–e_4, Fig. 6b) at the current turn are collected by querying the DEDB. These examples hold user templates (lexico-semantic utterances) which the natural language dialog system can understand. If these examples are too numerous to display, the system tries to select the appropriate example (e_4) by calculating the similarity of the current utterance (U_2) and the example utterance. We adopt this tactic because we assume that although the current utterance contains some errors, it also contains correct words.

3.4. Advantages of the EBDM

The overall strategy of the EBDM framework is executed through processes such as query generation, example search for retrieval, example selection for tie-breaking, and example recommendation for error recovery (Fig. 3). Our approach can be used for prototyping to produce dialog models for diverse applications. Although our dialog modeling approach to dialog management is conceptually simple and requires some corpus collection and annotation, it has several advantages for practical deployment considerations. First, the cost of our dialog modeling takes a small

Table 2
Goal-oriented dialog corpora statistics. #Dialog: number of dialog examples and #AvgUserTurn: average number of user turns.

Domain	#Dialog	#AvgUserTurn
Car navigation	120	4.28
Weather information	120	4.27
EPG	120	4.68

amount of corpus collection (Table 2). In fact, collecting dialog corpora is expensive because of human costs and it is often not practical to produce new dialogs for every demand. Even if dialogs can be automatically generated by user-system simulation, simulated dialogs do not replace the need to fully exploit the real data. To fulfill this requirement, we adopted a semantic-based indexing scheme, rather than a lexical pattern-based indexing scheme. We did this because the semantic-based indexing scheme requires a relatively small number of examples. In our approach, goal-oriented dialog systems can be developed given a fairly small number of annotated dialog corpora (fewer than 150 dialogs for each domain), whereas the RL-based dialog systems need many more dialog corpora (more than 10^3 dialogs) to automatically model dialog strategies.

The second advantage of our approach is that, in commercial system, the dialog policies can be maintained and modified by the demands of the users during real world deployment. In practical applications, stochastic approaches encounter obstacles when attempting to modify the learned policies because this modification requires a large number of dialog corpora. On the contrary, example-based approach allows for easy control and refinement of dialog flow by adding a small number of dialog examples.

The final advantage of our approach is that although the state variables of stochastic dialog modeling are often fairly limited by their complexity and search space, our approach can easily control the number of the state variables which contribute to the determination of the next system action. Our approach can determine state variables flexibly, by adding index constraints to build the DEDB. Subsequently, a system designer can empirically select and relax query constraints according to the domain-specific application.

4. Case studies for EBDM framework

We have used the EBDM framework to successfully build several spoken dialog systems, spanning different domains and levels of interaction complexity (Table 3). In this section, we discuss our EBDM-based dialog system in more detail.

4.1. Car navigation domain

We have developed a car navigation system using the EBDM framework. We first assembled a knowledge database of about 170 locations in Pohang city (South Korea) using web and tour guide books. The information includes location address, location type (e.g., restaurant, garage, gas station, hospital), location name, and phone number. In this system, interaction with the system starts with a greeting prompt. When the user provides necessary concepts to specify queries and determine the destination, the system generates appropriate actions on the devices controlled by the dialog system, such as searching the database to find matches to the user's queries. The system explicitly confirms whether its slot values is correct. In this domain, the user has an obvious goal to find the desired place and to start the navigation system. Thus, users behave in consistent and goal-directed ways to fill given slots; therefore, their utterances are highly correlated to the previous system utterances. In such transactional dialogs, the constraints of the discourse features are more important than any other constraints. Hence, the constraints for exact and partial matches always contain the discourse history vector and previous intentions.

4.2. Weather information domain

We also developed a system to give users easy access to weather forecast information in Korea, using natural language dialog. The information is automatically obtained from the Korean Meteorological Administration site[4] using the information extraction technology from the web pages. We extracted weather information for three days in more than 160 cities of South Korea. Weather information from the web site contains general weather conditions (e.g., sunny, partly cloudy), temperature ranges, and precipitation probabilities which are updated three times a day. In this domain, the user may have several requests for weather information of different cities or dates rather than selecting only one city or date. In such informational dialogs, the user utterances may not be highly correlated to the previous system utterances. The discourse features may be less important because the users need not provide all necessary information to fill the slots to query the database. Consequently, we do not consider the discourse history vector for partial match in this domain.

4.3. TV program guidance domain

With satellite and cable TV, the convergence of TV and Internet and the advent of digital networks, the number of TV channels is dramatically increasing. Thus, it is very difficult for the users to find their favorite programs quickly, and consequently, dialog systems for electronic program guide (EPG) service have become more popular. EPG service is an on-screen guide to scheduled broadcast TV programs, allowing a viewer to navigate, select, and discover content by time, title, channel, and genre. Therefore, we

[4] KMA web site: http://www.kma.go.kr/.

Table 3
Dialog systems implemented using EBDM framework. #Slots represents the number of domain-specific component slots used in each domain.

Domains	Descriptions	Genres	# Slots
Car navigation (CN)	Providing support for information access to city location database and selection of the desired destination in Pohang city	Transactional	6
Weather information (WF)	Providing information access service to weather forecast database in Korea	Informational	4
TV program guidance (EPG)	Providing electronic program guide (EPG) service to navigate, list, select content with TV schedule database in Korea	Informational	9
Chatbot (CB)	Providing daily conversation between human and computer for entertainment about 10 topics	Small talk	0

have developed a natural language dialog system to operate TV and support EPG services to find the name, genre, channel, cast, and time of TV programs in Korea. We automatically extracted EPG information of four public channels every week from the Korean EPG web site.[5] Users can record, set an alarm, and search their favorite TV programs in real time.

4.4. Chatbot

Our framework can support development of an entertainment chatbot to support interactions between human and computer (or intelligent robot). We can adapt the EBDM framework to support multi-domain chat dialogs as the following user utterances:

- I watched horror movies (movie).
- What kinds of sports do you like (sports)?
- It's too sunny (weather).

To semantically index the chat corpus, we use a tag set of the dialog acts because they are domain-independently defined. In this system, we can not consider semantic tags of the main goal or slot-filling information such as slot flags and discourse history vectors. We neglect this information because it is difficult for the current chatbot to extract general classes with high precision from user utterances. An exact match can be obtained using constraints of agent, domain and dialog act. For a partial query match, the dialog act is the only constraint of query keys due to its reliability, because the performance of dialog act prediction shows the best accuracy among other keys. We have summarized a set of constraints for exact and partial match depending on each application domain (Table 4).

Table 4
A set of constraints used in each dialog system. CN: car navigation domain; WI: weather information domain; EPG: electronic program guidance domain for TV; and CB: chatbot system. *X* means that we do not use the corresponding query constraint for the match type.

Constraints	Exact Match				Partial Match			
	CN	WI	EPG	CB	CN	WI	EPG	CB
AG								
DO								
DA								X
MG			X					X
SF			X		X	X	X	X
PREV_DA			X			X	X	X
PREV_MG			X			X	X	X
DHV			X			X	X	X

5. Multi-domain dialog system

We have investigated the development of multi-domain dialog systems using the EBDM framework. Because our system is based on extensible architecture similar to that developed by O'Neil et al. (2005), we can add and modify domain experts easily. Our multi-domain dialog system has double-layer system architecture in which the processes are divided into an expert selection step in the spotter modules and a dialog management step in the EBDM framework (Fig. 7). For the expert selection layer, we initially developed spotter modules which classify the target agent or domain of the current user utterance in the multi-domain dialog systems. For our system, we developed hierarchical spotting modules called agent spotter and domain spotter. At each turn, the user's utterance is identified either as chat or as a goal-oriented dialog. If the utterance is chat, it is assigned to a chat agent ($agent = chat$) for dialog processing. If it is goal-oriented dialog, it is assigned to a task agent ($agent = task$). Next, the domain spotter identifies the target domain of the current user utterance (e.g., *movie*, *sports*, *weather*).

For the spotter modules, we investigated a hybrid method of using keyword-based and feature-based approaches for automatically classifying the domains. In fact, the basic algorithm within the domain identification problem is similar to text categorization and topic detection. Many algorithms use the keyword-based spotting method and feature-based classification (Chelba et al., 2003; Komatani et al., 2006). Both the keyword spotting technique and the feature-based classification technique have still been used independently to develop the spotter module. In our system we define keyword features (e.g., *n*-best keyword and *n*-best domain class) which are extracted using the traditional keyword spotting method, and we use linguistic (e.g., part-of-speech (POS) tags and *n*-grams) and semantic features (e.g., dialog act) as well as the keyword features in the feature-based classification method (Table 5). To extract the keyword features, we

[5] EPG web site: http://www.epg.co.kr/.

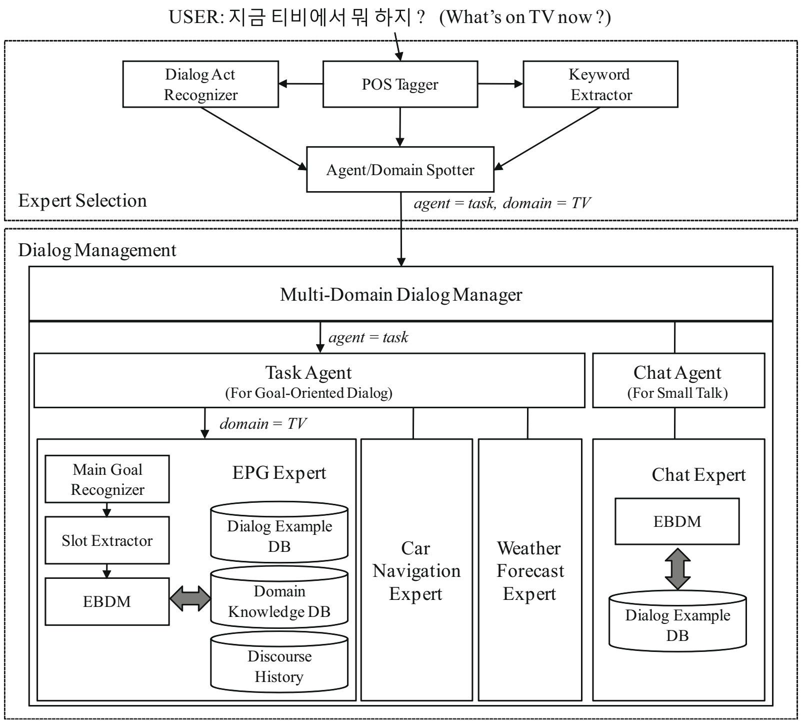


Fig. 7. System architecture for multi-domain dialog system.

Table 5
Features used in agent and domain spotter ($n \leqslant 2$).

Feature set	Description
Linguistic features	Word, POS tag, *n*-gram
Semantic features	Dialog act
Keyword features	*n*-Best keyword, *n*-best class (from TF ∗ IDF and salience weighting)

extract more obvious keywords which are assigned to the weight correlated to the particular domain class using the term weighting methods. For automatic extraction of keywords from the dialog corpora, we apply term frequency and inverse document frequency (TF ∗ IDF) (Salton and Yang, 1973) and salience measure (Gorin et al., 1997) as term weighting methods (Table 6). Although the linguistic features impose the keyword information using the bag-of-words, the keyword features are more indicative and reliable features than the pure bag-of-words to improve the performance of the domain identification.

Table 6
Partial examples of keywords. The domain class here represents the one maximally associated with the given keyword. A higher value means a greater correlation to the corresponding domain.

Keyword	TF ∗ IDF	Salience	Domain class
Drama	12.031	1.998	*TV*
Restaurant	6.197	0.435	*Navigation*
Rain	4.038	0.373	*Weather*
Lonely	2.173	0.155	*Love*
Football	9.831	2.297	*Sports*

6. Testing setup

6.1. Corpus collection

We collected the dialog corpora for goal-oriented dialogs and chat dialogs separately. For goal-oriented dialog systems, we employed ten people who knew how to operate the dialog system. They collected text-based human–human dialogs including about 1500 user utterances from 360 dialogs based on the wizard-of-Oz method (Fig. 8). We gave them a set of pre-defined scenarios related to domain-specific tasks (e.g., "Find a restaurant for lunch" in the car navigation domain). These human–human dialogs contain only reasonable examples because it is useless to manage goal-oriented dialogs if incorrect examples are selected. To collect the chat corpus containing about 2700 user utterances from 10 different domains, we used a web-based chatbot in Korea.[6] Ten people collected one-to-one examples (user-system turn pairs) from given domains by using text input. The DEDB for our chatbot holds domain-specific but goal-irrelevant dialogs for entertainment. For example, *self* domain class contains dialog examples about system's affair such as "*USER: What is your name?*" and "*USER: How do you do?*". In particular, we classified *tv* and *weather* domain class for both chat and

[6] Available at: http://www.aawoo.com.

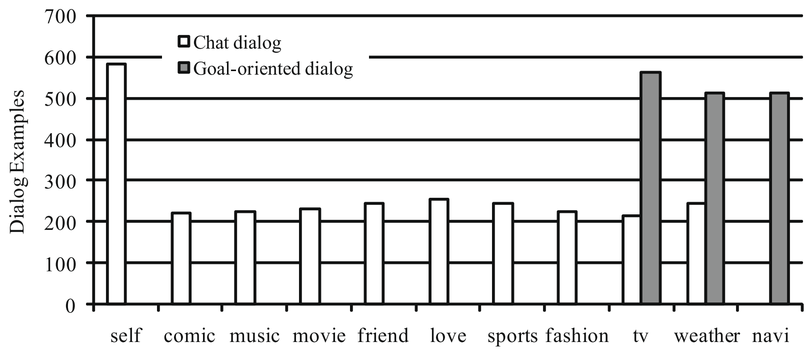


Fig. 8. Plot of the amount of dialog corpora on 11 domains.

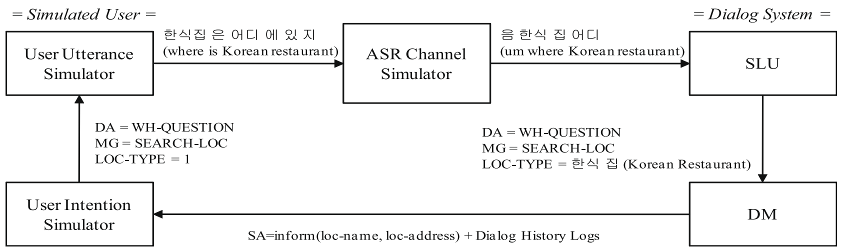


Fig. 9. Simulated environment for simulation evaluation.

goal-oriented dialogs.[7] All utterances were annotated for a language understanding task given a tag set of dialog acts, main goals, and domain-specific component slots. We also annotated the system actions manually for the EBDM. Subsequently, these corpora were used to train user simulator, SLU, agent/domain spotter, and dialog models. To automatically evaluate the dialog systems, PARADISE framework has been widely known as a general framework for automatic estimation of subjective user satisfaction from objective dialog performance measures using multivariate linear regression (Walker et al., 1997). However, in this study, we simply use objective dialog performance measures such as TCR and its average user turn length to evaluate the dialog systems.

6.2. Dialog simulator

To evaluate our dialog models by simulating a realistic scenario, we developed a simulated environment with a user simulator and an ASR channel (Fig. 9) (Jung et al., 2008). The statistical user simulator is composed of two components: an *Intention Simulator* and a *Surface Simulator*. First the *Intention Simulator* is implemented by using the conditional random fields (CRFs) model to consider the dialog sequences as a sequential labeling problem. The simulator can calculate the probability of the next user intention from the current discourse context features. To generate diverse user intentions, we randomly selected the user intention based on the probability distribution of the user intention given the discourse context. After selecting the user intention, the *Surface Simulator* was used to generate a user utterance to express the selected intention using intention-specific POS tag sequences and dictionaries. Then, the ASR channel was used to automatically transform 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 substitutions, deletions, and insertions. The substitution candidates were sampled by phoneme-level sequence alignment using the phoneme confusion matrix. Finally, the one-best ASR hypothesis was selected to maximize the language model score among sampled utterances.

To automatically measure the quality or success of the generated dialogs, we defined the final states of 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). If a dialog flow reaches the final state, the evaluator regards that dialog's task as successfully completed. Unfortunately this assessment is possible only when the goal of the dialog is explicitly identified. Therefore, in simulated user evaluation, we evaluated the car navigation system because this task explicitly shows transactional flows in which the users have the obvious domain-specific goal of selecting the final destination, and we excluded other tasks.

6.3. Dialog systems

We developed two car-navigation dialog systems by two different methods, and compared the systems to the one

[7] In this paper, *EPG* and *weather information* represent goal-oriented dialogs for *tv* and *weather* domain, respectively.

developed using our EBDM approach. As a baseline system, we created a hand-crafted (*HC*) dialog system which is represented as a finite-state controller consisting of nodes and directed edges (Fig. 10). Each node includes four different components: (1) a precondition that must be true before the subtask is executed; (2) a description of the node that includes its label and identifier; (3) links to nodes that may be executed at the subsequent turn; and (4) the system actions to manage the current turn. Each node corresponds to a user goal, and each edge denotes a possible transition from the current state to the next. If the current input matches any precondition of the user goal, the corresponding system action is selected to complete the current turn (Fig. 11). This system does not contain any dialog examples to determine the next system action; this characteristic is distinct from the EBDM framework. The task in this domain is to provide information and navigation for desired destination. The dialog system first takes the action *greet*. Users may select specific values for three different slots: location address, location type, and location name. They may also then ask for information about the phone number of the location, and they select as the route type (fastest path or easiest path). If all slots are successfully filled, the system requests that the user explicitly confirm each slot to determine the destination (e.g., *SYSTEM: You selected Country Food of Korean Restaurant in Daeyi-dong by the easiest path. Is this correct?*). If the users provide inconsistent information, the system treats the new information as correct and confirms the new information again. Finally, the users ask to show the final path from here to a specific destination and the system takes the action *guide* to indicate the path.

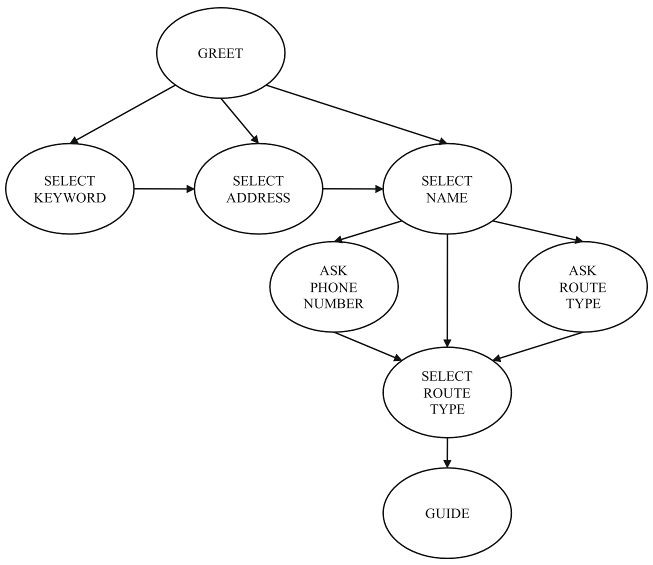
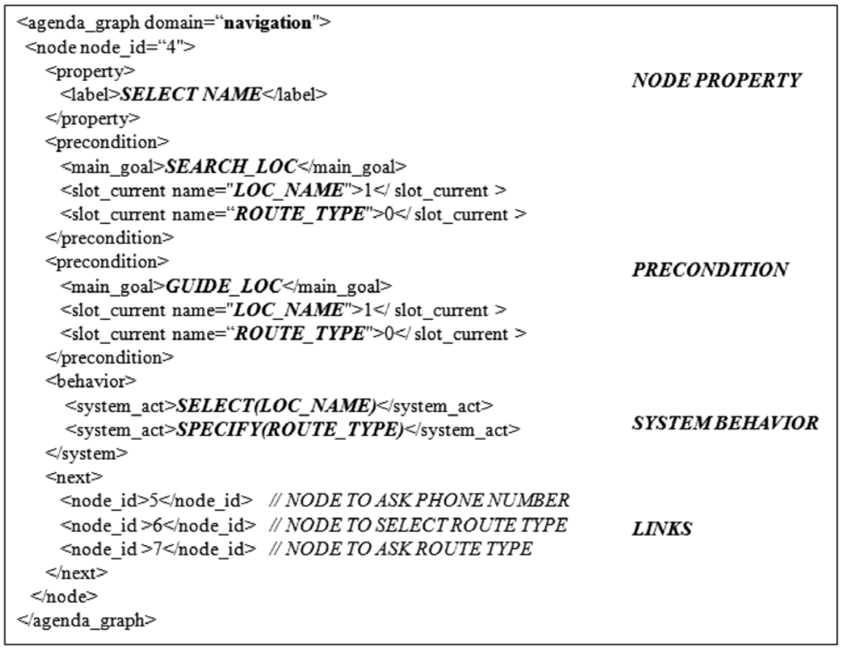


Fig. 10. Logical diagram of hand-crafted strategy (*HC*) on car navigation domain.

The second dialog system was a variant of the EBDM framework that we developed using the BLEU measure (Papineni et al., 2002) (EBDM–BLEU) to calculate the utterance similarity instead of edit distance. The BLEU measure is a widely-used text similarity measure for automatic evaluation in the machine translation (MT) research community. It was implemented and applied to the lexico-semantic similarity for selecting the best dialog example. We used the BLEU-3 measure, which is calculated from unigram to trigram, because most utterances in the dialog corpora are shorter than the sentences in the MT task.


```
<agenda_graph domain="navigation">
 <node node_id="4">
   <property>                                              NODE PROPERTY
     <label>SELECT NAME</label>
   </property>
   <precondition>
     <main_goal>SEARCH_LOC</main_goal>
     <slot_current name="LOC_NAME">1</ slot_current >
     <slot_current name="ROUTE_TYPE">0</ slot_current >
   </precondition>
   <precondition>                                          PRECONDITION
     <main_goal>GUIDE_LOC</main_goal>
     <slot_current name="LOC_NAME">1</ slot_current >
     <slot_current name="ROUTE_TYPE">0</ slot_current >
   </precondition>
   <behavior>
      <system_act>SELECT(LOC_NAME)</system_act>
      <system_act>SPECIFY(ROUTE_TYPE)</system_act>        SYSTEM BEHAVIOR
   </system>
   <next>
     <node_id>5</node_id>   // NODE TO ASK PHONE NUMBER
     <node_id >6</node_id>  // NODE TO SELECT ROUTE TYPE   LINKS
     <node_id >7</node_id>  // NODE TO ASK ROUTE TYPE
   </next>
  </node>
</agenda_graph>
```
Fig. 11. Example of XML rules for HC dialog strategy.

7. Evaluation

7.1. Domain identification evaluation

Before evaluating the dialog model, we verified that our spotters correctly predict the target agent/domain of each utterance (Table 7). The performance of the proposed spotting technique was evaluated for combination of feature sets applying the maximum entropy (ME) classifier because it has shown good performance in various areas of natural language processing (Berger et al., 1996). The performance of the agent/domain spotter was evaluated by the feature sets with the ME classifier. These were evaluated using McNemar's test with the data that were randomly divided into a training set (3424 utterances) and a test set (856 utterances) (Dietterich, 1998).

The agent spotter module seems to have excessively high accuracy (97.20%; Table 7) when using only the linguistic features as a baseline performance because this is relatively easy binary classification task. However, each class (*chat and task*) consists of different topics, and addition of keyword features did not significantly improve performance ($p > 0.01$). However, although the domain spotter module has to solve a multi-class classification problem (11 classes), it yielded the best accuracy of 89.72% with TF $*$ IDF weighting which was a statistically significant improvement ($p < 0.01$). This result shows that the use of keyword features is helpful to improve the performance of domain identification and that our spotter modules are capable of detecting the agent and domain. Therefore, these modules allow smooth implicit switching in multi-domain dialog systems.

7.2. Simulated user evaluation

First, we explored the correlation between WER and concept error rate (CER) to verify our ASR channel (Fig. 12a). Our ASR channel can models speech recognition errors as well as SLU errors up to a certain error rate. We also tested the robustness of the proposed dialog systems according to WER values. TCRs and #AvgUserTurn were measured under various WER conditions ranging from 0% to 50% (Fig. 12b). The dialog systems could be evaluated with a relatively large number of simulated dialogs in the simulated user evaluation whereas in the real user evaluation only a few dialogs could be evaluated.

Table 7
Performance of agent/domain spotter.

Feature set		Agent spotter (%)	Domain spotter (%)
Baseline (only linguistic features)		97.20	84.78
+ Semantic features		97.90	86.37
+ Keyword features	TF $*$ IDF	98.37	89.72
	Salience	98.16	87.91

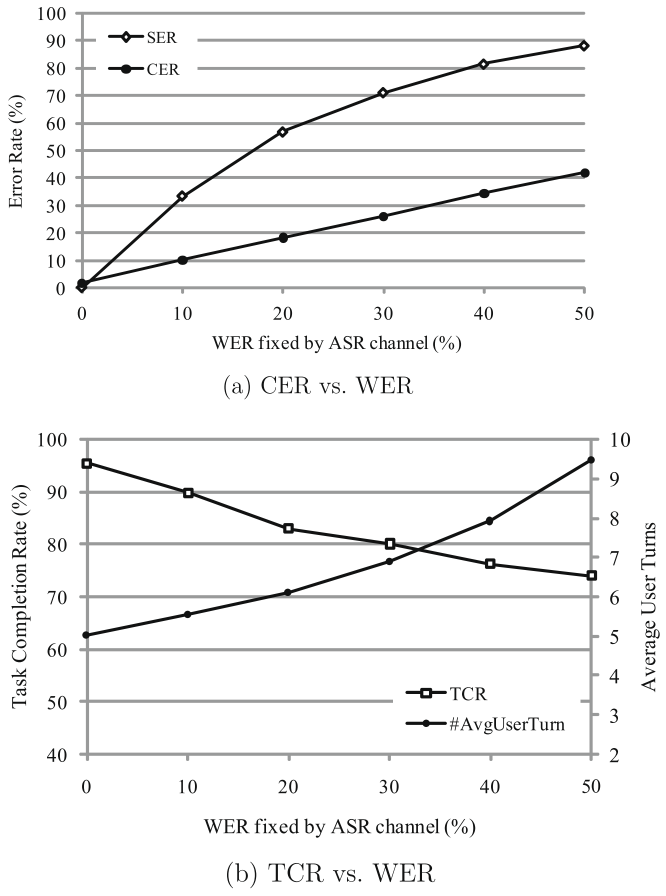


Fig. 12. Sentence error rate (SER), concept error rate (CER), task completion rate (TCR), and average user turns (#AvgUserTurn) under various word error rate (WER) conditions.

Three thousand simulated dialogs were evaluated for each system at each WER condition (Fig. 12). Our approach successfully models dialog strategies, and can tolerate speech recognition errors to a certain extent, although an increasing error rate increases the average number of user turns (#AvgUserTurn).

We next performed empirical analysis to determine the optimal interpolation weight α (Eq. (1)) to compute the utterance similarity measure (Fig. 13). For example, $\alpha = 0$ corresponds to the case in which no lexico-semantic patterns are used to compute the utterance similarity. For WER = 0%, the best TCR was achieved when $\alpha = 0$. However, when WER was 20%, the highest TCR was achieved at $\alpha = 0.1$, and when WER was 50%, the highest TCR was achieved at $\alpha = 0.3$. This simulation result is useful to determine the optimal weight α. Even if real developers have no user simulator, they can use intuition to determine α according to the characteristics of dialogs and the performance of ASR module. For example, if the task of the dialogs is a slot-filling problem such as a car navigation task, the developer can select a lower α to reflect the higher S_{DHS} for the slot-filling vector. In addition, if the performance of ASR system is not good, the developers can select a relatively small α to achieve robust dialog management by

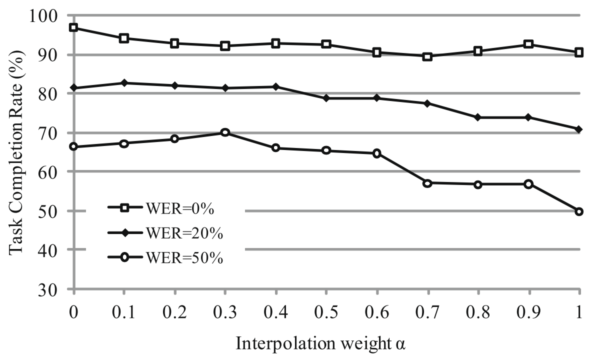


Fig. 13. Optimal weight for utterance similarity computation. Values on x-axis denotes the weight α (Eq. 1).

rejecting erroneous utterances, because the out-of-coverage patterns which could be erroneous inputs may cause ASR and SLU errors. However, if α is too high, most inputs are rejected as out-of-coverage patterns due to ASR and SLU errors, and then TCRs are decreased by these false rejections. For example, the worst TCRs occurred at $\alpha \simeq 1$ (Fig. 13).

We also explored how many dialog examples are necessary to develop a car navigation system. We randomly chose and indexed the dialog examples in the dialog corpus. WER was set to 10% because the performance of the current ASR system is usually better than this value. With only 450 examples used in the DEDB, the TCR converged to about 90% and #AvgUserTurn to about 5.5 turns (Fig. 14). This result verifies that the EBDM framework does not require a large corpus for developing goal-oriented dialog systems, because typical users show consistent and goal-directed inquiry patterns when using goal-oriented dialog systems.

Finally, we compared our EBDM method with the hand-crafted dialog system as a baseline system. Three thousand simulated dialogs were evaluated for each system (Table 8). The EBDM frameworks were compared to HC at a variety of WERs. Both EBDM–ED and EBDM–BLEU significantly outperform the HC system when

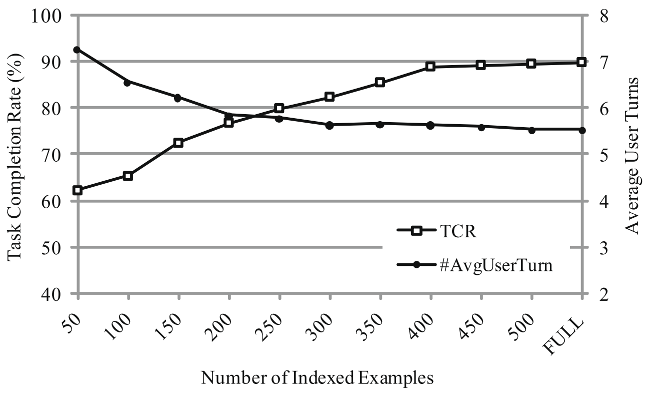


Fig. 14. #AvgUserTturn and TCR vs. number of indexed examples (WER = 10%). A label of "FULL" on x-axis means that the full set of dialog examples are used to create the example database.

ASR errors occurred frequently (WER > 30%). This result shows that the EBDM frameworks are robust to recognition errors. One reason for this is that our approach can also take into account a flexible set of semantic and discourse constraints by using the relaxation strategy, whereas HC is restricted to examining condition to take action based on the previous state. In addition, our approach can reduce error propagation by rejecting erroneous inputs at the example selection step. The effect of text similarities was also investigated by varying WERs. Higher TCR and shorter #AvgUserTurn are achieved when more recognition errors are made especially in the EBDM–BLEU system. The reason for this result is that the edit distance measure uses sequence alignment in which the word order is important, but the BLEU uses position-independent n-gram matching. Insertion and deletion errors may be more critical to the sequence alignment problem rather than to the matching problem. Although the edit distance is more rigorous to accept scrambled inputs, word order is more flexible in Korean than in English. Thus, EBDM–BLEU can be more robust in noisy environments even if insertion and deletion of stop words is triggered by ASR module.

7.3. Real user evaluation

To evaluate our approach in a real situation, we employed 10 undergraduates from the Pohang University of Science and Technology (POSTECH). We provided them with pre-defined subjects and asked them to collect test data of about 760 user utterances which were independent of the dialog corpora previously used in developing the system. These data were collected assuming that the test users were in cars and were accessing information using all-in-one mobile devices. After processing each dialog, the participants completed a questionnaire about successful turn and task completion to assess their satisfaction with aspects of the performance evaluation. We evaluated TCR and success turn rate (STR) for each dialog system. STR designates the average success turn rate of the system response for every user utterance. Therefore, STR reflects the utterance-level accuracy of the system action evaluated by real users. We did not evaluate the TCR of the chatbot system in which the users do not usually have domain-specific goals but only communicate for entertainment. The best TCR was 94% in the weather information domain (Table 9) because this domain has relatively low interaction complexity and a small number of component slots. In the multi-domain dialog system, #AvgUserTurn was higher and TCR was lower than those of single-domain dialog systems because new errors occurred in the spotter modules. Although these results may not be significant due to the small size of test data, we believe our approach will be useful for prototyping diverse single or multi-domain applications including goal-oriented dialogs and chat dialogs.

In addition to the performance test, we performed an error analysis of our systems by examing the dialog logs.

Table 8
Evaluation of the EBDM framework by using edit distance (EBDM–ED) and BLEU (EBDM–BLEU) measures, and hand-crafted strategy (*HC*) in a car navigation domain.

Systems	Metrics	WER (%)					
		0	10	20	30	40	50
HC	CER (%)	1.41	9.06	16.19	23.71	31.04	39.51
	TCR (%)	96.58	90.50	84.72	73.43	59.00	46.73
	#AvgUserTurn	5.65	6.99	8.36	9.90	11.70	13.61
EBDM–ED	CER (%)	2.19	9.56	16.30	23.60	31.73	39.18
	TCR (%)	97.39	91.16	86.63	80.80	73.67	66.33
	#AvgUserTurn	4.92	5.37	5.91	6.42	7.48	9.07
EBDM–BLEU	CER (%)	2.56	9.95	17.93	25.62	33.01	40.08
	TCR (%)	96.78	91.06	83.22	82.51	76.38	70.85
	#AvgUserTurn	4.88	5.29	5.83	6.25	7.22	8.64

Table 9
Success turn rate (STR) and task completion rate (TCR) of each dialog system by human evaluation (WER = 0%).

System	#Dialog	#AvgUserTurn	STR (%)	TCR (%)
Car navigation	50	4.54	86.25	92.00
Weather information	50	4.46	89.01	94.00
EPG	50	4.50	83.99	90.00
Chatbot	50	5.60	64.31	–
Multi-domain	15	6.08	78.77	86.67

First we measured the coverage of the dialog examples using the example match rate which denotes the average match rate of the query to the DEDB for each case (exact match, partial match, and no example) (Table 10). During the goal-oriented dialogs, almost all utterances were exactly or partially matched to one of the dialog examples belonging to the DEDB because there appear to be strong correlations between system prompts and typical user answers. However, *No Example* errors occurred frequently during the chat dialogs, because it is very difficult to cover all possible patterns of the chat dialogs with a small number of dialog examples. This finding reflects that the users give more diverse inputs to the chatbot system than to the other systems.

We have also analyzed why the *No Example* errors occurred in our systems. One of the major reasons is that most errors were caused by misrecognizing and misunderstanding errors in which the current semantic frame contains incorrect prediction results from SLU module. Another reason is that the data sparseness ca cause unexpected errors even if the SLU and ASR results are correct. Our approach cannot handle patterns that are significantly different from patterns stored in the DEDB. For this case, our goal-oriented dialog systems reject the current input and give helpful feedback messages for rephrasing at the example recommendation step. We think that this strategy can improve the usability of the goal-oriented dialog systems for practical deployment by circumventing misunderstanding errors and the data sparseness problem.

Table 10
Example match rate (EMR) of each dialog system by human evaluation (%).

System	Exact match	Partial match	No example
Car navigation	50.22	44.49	5.29
Weather information	69.49	25.00	5.51
EPG	58.33	37.22	4.45
Chatbot	50.71	14.29	35.00
Multi-domain	69.23	24.62	6.15

8. Conclusion and discussion

In this paper, we described example-based dialog modeling (EBDM), a generic dialog modeling scheme for natural language dialog systems that can operate in multiple applications. EBDM uses semantic-based indexing and querying using a set of flexible constraints (e.g., domain, dialog act, main goal, and previous intention). The dialog manager can predict the next system action based on the best similar example, by using an utterance similarity measure. We also adapted an error recovery strategy to handle *No Example* errors that result from unexpected input or error propagation. When the dialog manager in our system cannot find similar examples, it triggers an *UtterHelp* strategy which requests that the users restate their instruction feedback for the users. We have presented several case studies for developing applications using EBDM. Our evaluation results demonstrate that use of example-based dialog systems has great promise in natural language dialog systems. In addition, we have investigated use of a hybrid approach for spotting modules to determine the domain of a given dialog using feature-based classification augmented by the keyword spotting method. The result shows that this hybrid approach is useful for domain identification.

There are some challenges for future research on EBDM. One involves the uncertainty of the input that can be caused by recognition or understanding errors to search candidate examples and to select the best example. This uncertainty of the state may be resolved by using a domain-specific agenda or task model to search and select examples given *n*-best recognition hypotheses. We are also currently working on transferring the EBDM framework to a new application (e.g., English tutoring and multi-modal dialog systems). This system will be developed as affective interfaces based on user context such as skill level, hastiness, and emotion.

Acknowledgement

This work was supported by Grant No. RTI04-02-06 from the Regional Technology Innovation Program and by the Intelligent Robotics Development Program, one of the 21st Century Frontier R&D Programs funded by the Ministry of Knowledge Economy (MKE).

Appendix A. Sample conversations with the EBDM framework

This appendix contains several sample conversations with our dialog systems: car navigation, EPG, weather information, and multi-domain dialog system.

A.1. Conversation with car navigation system

(1) S: Hi. This is the car navigation system for Pohang city. How may I help you?
(2) U: Is there any restaurant?
(3) S: There are some; in Daeyidong, Hyojadong, Snagdaedong, and Duhodong.
(4) U: How about Korean restaurants?
(5) S: There is Country Food in Daeyidong, Following Plain in Daeyidong, Sweet Room in Hyojadong, and Rice Soup in Duhodong.
(6) U: Then, let me go to Country Food in Daeyidong.
(7) S: Ok. You selected Country Food in Daeyidong. Choose the route type of the fastest or the easiest path.
(8) U: The fastest path, please.
(9) S: Ok. You selected the fastest path. You have selected Country Food in Daeyidong as the fastest path. Is it correct?
(10) U: Yes.
(11) S: Then, I will show the route to Country Food in Daeyidong.

A.2. Conversation with EPG system

(1) S: Hi. This is the EPG system. How may I help you?
(2) U: Turn on TV, please.
(3) S: TV is on. It is 9 PM.
(4) U: What's on TV now?
(5) S: "MBC news desk" is on at MBC, "Salt doll" at SBS, "KBS 9'o clock News" at KBS1, "The Briefing of Entertainment World" at KBS2. Please select a favorite channel.
(6) U: Then, when do movies start?
(7) S: There is no movie program. Please select other genres.
(8) U: Who is the cast of Salt doll?
(9) S: The casts of Salt doll are Soojung, Hwang and Jiseok, Kang.
(10) U: Ok. Let me record it.
(11) S: Yes. I will record "Salt doll".

A.3. Conversation with weather information system

(1) S: Hi. This is the weather information system in Korea. How may I help you?
(2) U: Can you tell me what the weather is like in Seoul today?
(3) S: You selected the weather in Seoul today. Can I tell you this?
(4) U: Yes.
(5) S: In Seoul this afternoon, it will be partly sunny early, then becoming cloudy.
(6) U: What about Pohang?
(7) S: You selected the weather in Pohang. Can I tell you the weather today?
(8) U: No, tomorrow.
(9) S: In Pohang tomorrow, it will be sunny.
(10) U: What's the temperature range?
(11) S: You selected the temperature range. Can I tell you the temperature range in Pohang tomorrow?
(12) U: Yes.
(13) S: In Pohang, the highest is 32 °C and the lowest is 26 °C.

A.4. Conversation with multi-domain dialog system

(1) S: You can ask about car navigation, EPG, weather information, or chat. How may I help you? (GREETING)
(2) U: Hi. (*agent* = *chat*, *domain* = *self*)
(3) S: Hello. Nice to meet you.
(4) U: Can you tell me what the weather is like in Pohang today? (*agent* = *task*, *domain* = *weather*)
(5) S: You selected the weather in Pohang today. Can I tell you this?
(6) U: Yes. (*agent* = *task*, *domain* = *weather*)
(7) S: In Pohang this afternoon, it will be sunny.
(8) U: It's nice day. (*agent* = *chat*, *domain* = *weather*)
(9) S: Let's go on a picnic!
(10) U: I want to go to Country Food in Daeyidong. (*agent* = *task*, *domain* = *navigation*)
(11) S: Ok. You selected Country Food in Daeyidong. Choose the route type of the fastest or the easiest path.
(12) U: The easiest path, please. (*agent* = *task*, *domain* = *navigation*)
(13) S: Ok. You selected the easiest path. You have selected Country Food in Daeyidong as the easiest path. Is it correct?
(14) U: Sure. (*agent* = *task*, *domain* = *navigation*)

(15) S: Then, I will show the route to Country Food in Daeyidong.

References

Allen, J., Byron, D., Dzikovska, M., Ferguson, G., Galescu, L., Stent, A., 2000. An architecture for a generic dialogue shell. Nat. Language Eng. 6 (3), 1–16.

Berger, A., Della Pietra, V., Della Pietra, S., 1996. A maximum entropy approach to natural language processing. Comput. Linguist. 22 (1), 39–71.

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

Bui, T., Rajman, M., Melichar, M., 2004. Rapid dialogue prototyping methodology. In: Proc. Internat. Conf. on Text, Speech, and Dialogue, pp. 579–586.

Chelba, C., Mahajan, M., Acero, A., 2003. Speech utterance classification. In: Proc. IEEE Internat. Conf. on Acoustics, Speech and Signal Processing, pp. 69–72.

Dietterich, T.G., 1998. Approximate statistical tests for comparing supervised classification learning algorithms. Neural Comput. 10, 1895–1923.

Eun, J., Jeong, M., Lee, G., 2005. A multiple classifier-based concept-spotting approach for robust spoken language understanding. In: Proc. Eur. Conf. on Speech, Communication and Technology, pp. 3441–3444.

Georgila, K., Henderson, J., Lemon, O., 2005. Learning user simulations for information state update dialogue systems. In: Proc. Eur. Conf. on Speech, Communication and Technology, pp. 893–896.

Gorin, A., Riccardi, G., Wright, J., 1997. How may I help you? Speech Comm. 23, 113–127.

Henderson, J., Lemon, O., Georgila, K., 2005. Hybrid reinforcement/supervised learning for dialogue policies from COMMUNICATOR data. In: Proc. Workshop on Knowledge and Reasoning in Practical Dialogue Systems, International Joint Conference on Artificial Intelligence.

Hurtado, L.F., Griol, D., Sanchis, E., Segarra, E., 2005. A stochastic approach to dialog management. In: Proc. IEEE Workshop on Automatic Speech Recognition and Understanding, pp. 226–231.

Inui, M., Ebe, T., Indurkhya, B., Kotani, Y., 2001. A case-based natural language dialogue system using dialogue act. In: Proc. IEEE Internat. Conf. on Systems, Man, and Cybernetics, pp. 193–198.

Jenkins, M.-C., Churchill, R., Cox, S., Smith, D., 2007. Analysis of user interaction with service oriented chatbot systems. In: Proc. Internat. Conf. of Human–Computer Interaction, pp. 76–83.

Jung, S., Lee, C., Kim, K., Lee, G.G., 2008. An integrated dialog simulation technique for evaluating spoken dialog systems. In: Proc. Workshop on Speech Processing for Safety Critical Translation and Pervasive Applications, International Conference on Computational Linguistics, pp. 9–16.

Komatani, K., Kanda, N., Nakano, M., Nakadai, K., Tsujino, H., Ogata, T., Okuno, H.G., 2006. Multi-domain spoken dialogue system with extensibility and robustness against speech recognition errors. In: Proc. 7th SIGDIAL Workshop on Discourse and Dialogue, pp. 9–17.

Lamel, L., Rosset, S., Gauvain, J., Nennacef, S., 1999. The LIMSI ARISE system for train travel information. In: Proc. IEEE Internat. Conf. on Acoustics, Speech and Signal Processing, pp. 501–504.

Larsson, S., Ericsson, S., 2002. GoDiS – issue-based dialogue management in a multi-domain, multi-language dialogue system. In: Demo Abstract in the Association of Computational Linguistics, pp. 104–105.

Larsson, S., Traum, D.R., 2006. Information state and dialogue management in the TRINDI dialogue move engine toolkit. Nat. Language Eng. 6, 323–340.

Lee, C., Jung, S., Eun, J., Jeong, M., Lee, G., 2006. A situation-based dialog management using dialog examples. In: Proc. IEEE Internat. Conf. on Acoustics, Speech and Signal Processing, pp. 69–72.

Lemon, O., Georgila, K., Henderson, J., Stuttle, M., 2006. An ISU dialogue system exhibiting reinforcement learning of dialogue policies: generic slot-filling in the TALK in-car system. In: Proc. Eur. Chapt. of Association of Computational Linguistics, pp. 119–122.

Lemon, O., Gruenstein, A., Peters, S., 2002. Multi-tasking and collaborative activities in dialogue systems. In: Proc. 3rd SIGDIAL Workshop on Discourse and Dialogue, pp. 113–124.

Lesh, N., Rich, C., Sidner, C., 2001. Collaborating with focused and unfocused users under imperfect communication. In: Proc. Internat. Conf. 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 Trans. Speech Audio Process. 8, 11–23.

Lopez-Cozar, R., la Torre, A.D., Segura, J.C., Rubio, A.J., 2003. Assessment of dialogue systems by means of a new simulation technique. Speech Comm. 40 (3), 387–407.

McTear, M., 1998. Modelling spoken dialogues with state transition diagrams: experience of the CSLU toolkit. In: Proc. Internat. Conf. on Spoken Language Processing, Vol. 4, pp. 1223–1226.

Minker, W., Haiber, U., Heisterkaml, P., Scheible, S., 2004. SENECA spoken language dialogue system. Speech Comm. 43, 89–102.

Murao, H., Kawaguchi, N., Matsubara, S., Ymaguchi, Y., Inagaki, Y., 2003. Example-based spoken dialogue system using WOZ system log. In: Proc. 4th SIGDIAL Workshop on Discourse and Dialogue, pp. 140–148.

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

O'Neil, I., Hanna, P., Liu, X., Greer, D., McTear, M., 2005. Implementing advanced spoken dialogue management in Java. Speech Comm. 54, 99–124.

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

Pakucs, B., 2003. Towards dynamic multi-domain dialogue processing. In: Proc. Eur. Conf. on Speech, Communication and Technology, pp. 741–744.

Papineni, K., Roukos, S., Ward, T., Zhu, W.-J., 2002. BLEU: a method for automatic evaluation of machine translation. In: Proc. Assoc. for Computational Linguistics, pp. 311–318.

Peckham, J., 1993. A new generation of spoken dialog systems: results and lessons from the SUNDIAL project. In: Proc. Eur. Conf. on Speech, Communication and Technology, pp. 33–40.

Rich, C., Sidner, C., 1998. Collagen: a collaboration agent for software interface agents. J. User Model. User-Adapt. Interact. 8 (3), 315–350.

Salton, G., Yang, C., 1973. On the specification of term values in automatic indexing. J. Docum. 29, 351–372.

Schatzmann, J., Georgila, K., Young, S., 2005. Quantitative evaluation of user simulation technique for spoken dialogue systems. In: Proc. 6th SIGDIAL Workshop on Discourse and Dialogue, pp. 45–54.

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

Thomson, B., Schatzmann, J., Young, S., 2008. Bayesian update of dialogue state for robust dialogue systems. In: Proc. IEEE Internat. Conf. on Acoustics, Speech and Signal Processing, pp. 4937–4940.

Walker, M., Aberdeen, J., Boland, J., Bratt, E., Garofolo, J., Hirschman, L., Le, A., Lee, S., Narayanan, S., Papineni, K., Pellom, B., Polifroni, J., Potamianos, A., Prabhu, P., Rudnicky, A., Sanders, G., Seneff, S., Stallard, D., Whittaker, S., 2001. DARPA communicator dialog travel planning systems: the June 2000 data collection. In: Proc. Eur. Conf. on Speech, Communication and Technology, pp. 1371–1374.

Walker, M.A., Litman, D.J., Kamm, C.A., Abella, A., 1997. PARADISE: a framework for evaluating spoken dialogue agents. In: Proc. Eur. Chapt. Assoc. of Computational Linguistics, pp. 271–280.

Watanabe, T., Araki, M., Doshita, S., 1998. Evaluating dialogue strategies under communication errors using computer-to-computer simulation. IEICE Trans. Inform. System E81-D (9), 1025–1033.

Weng, F., Varges, S., Raghunathan, B., Ratiu, F. Pon-Barry, H., Lathrop, B., Zhang, Q., Scheideck, B., Xu, K., Purver, M., Mishra, R., Lien, A., Raya, M., Peters, S., Meng, Y., Russell, J., Cavedon, L., Shriberg, E., Schmidt, H., Prieto, R., 2006. CHAT: a conversational helper for automotive tasks. In: Proc. Internat. Conf. of Spoken Language Processing, pp. 1061–1064.

Williams, J., Young, S., 2007. Partially observable markov decision processes for spoken dialog systems. Comput. Speech Language 21, 393–422.

Williams, J.D., Young, S., 2005. Scaling up POMDPs for dialog management: the "Summary POMDP Method". In: Proc. IEEE Workshop on Automatic Speech Recognition and Understanding, pp. 250–255.

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

Zue, V., Seneff, S., Glass, J., Polifroni, J., Pao, C., Hazen, T., Hetherington, L., 2000. JUPITER: a telephone-based conversational interface for weather information. IEEE Trans. Speech Audio Process. 8, 85–96.