EMERGENT COMPUTATIONAL DIALOGUE MANAGEMENT ARCHITECTURE FOR TASK-ORIENTED SPOKEN DIALOGUE SYSTEMS

Takeshi Kawabata

NTT Basic Research Laboratories
3-1 Morinosato-Wakamiya, Atsugi-shi 243-0198, JAPAN
http://www.brl.ntt.co.jp/info/dug/index.html
kaw@idea.brl.ntt.co.jp

ABSTRACT

This paper proposes a new dialogue management architecture for human-machine speech communication systems. In our daily speech communication, incremental, non-deterministic and quick-response behaviors are required for effortless information interchange. Emergent computational architectures, proposed in the robot control domain, are promising to enable such features. The dialogue manager (ECL-DIALOG) consists of multiple "phrase pattern" detectors as input sensors. The CFG driven phrase detectors search for phrase patterns in user utterances and generates numerous emergent slot-filling signals. The system integrates them according to their "phrase pattern" priorities and updates the current task-completion context. When a slot value is updated, the system generates an appropriate response. For example, when the system finds a new slot value from user utterances, the system generates a chiming utterance "yeah". When the context slot is replaced by a different value "Tuesday" that has a lower priority, the system asks for confirmation "On Tuesday?".

1. INTRODUCTION

Speech communication is a promising medium for constructing an effortless human-machine information interchange system. However, up-to-date speech recognition systems could not achieve practical recognition accuracy yet. On the contrary, nearly 100% task completion is accomplished in human-to-human communication by applying dialogue coordination and confirmation behaviors. A well-designed dialogue manager may compensate for speech recognition accuracy by applying such behaviors.

Traditional dialogue management architectures often employ state transition models (automata). Each state of the automaton binds speech recognition constraints, a vocabulary set and sentence patterns. The system recognizes user utterances corresponding to each dialogue

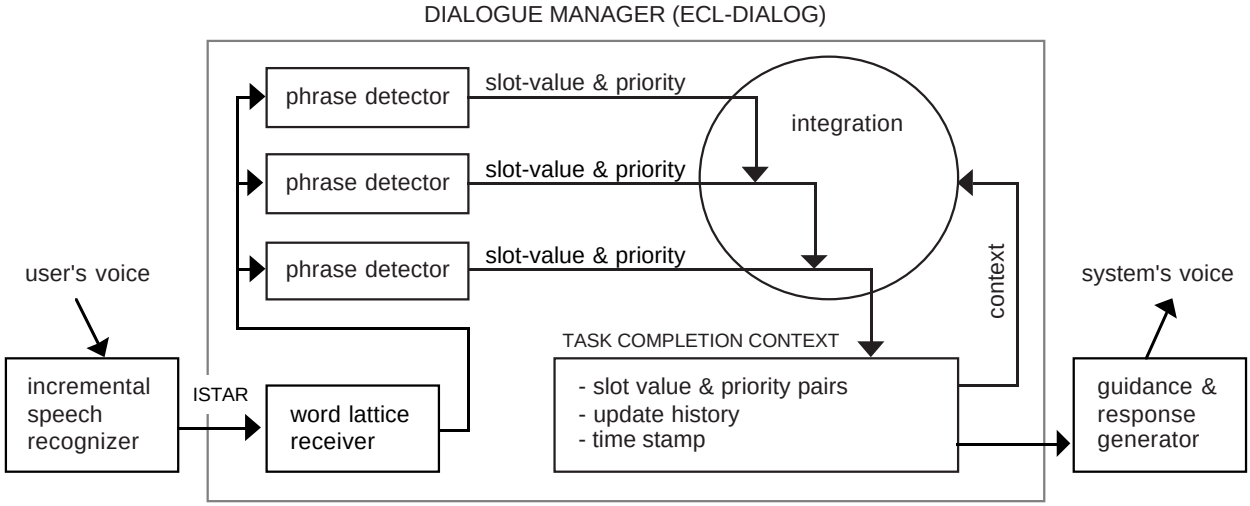


Fig. 1 Schematic Diagram of an Emergent Computational Spoken Dialog System

scene under the current state constraints. The state also has transition paths to subsequent states. The system determines the next state according to the recognition result. Unfortunately, such a deterministic model does not work well in a real spoken dialog system. The speech recognizer makes mistakes. Users may utter out-of-vocabulary words. These inappropriate words may cause incorrect state transitions and result in a fatal conflict of dialogue scene identification between a user and the system. For example, when the system expects a direction word, the user utterance "No, thank you" may be misinterpreted as "Northern". The misinterpretation leads to other incorrect transitions. So the situation becomes worse and worse.

One solution of this problem is a non-deterministic approach [1]. A stochastic distributed state-transition model can avoid fatal errors [2]. However, this method needs much calculation amounts. It is not suitable for a real time spoken dialogue system.

Emergent computational architectures were proposed in the robot control domain [3]. Multiple decision mechanisms, which have sensor inputs and motor controllers, are hierarchically and concurrently arranged. Usually, higher-level decision mechanisms control the robot behaviors. However, during an emergent crisis, lower-level decision mechanisms take quick actions. This mechanism is promising for the construction of an incremental, non-deterministic quick-response dialogue management system.

This paper describes an emergent computational dialog management architecture. The target of our system is to get appropriate slot values for task completion from user utterances through interactive dialogues. Quite a few practical applications result from such task-oriented slot filling semantics.

2. ECL-DIALOG

Figure 1 shows the schematic diagram of the spoken dialogue system based on the Emergent Computational Language processing. The system consists of three modules, a speech recognizer, dialogue manager and response generator. This paper describes them in order.

The system is currently implemented for the “meeting-room appointment” task in Japanese. We explain some parts of them in English for reader convenience.

2.1 Speech Recognition

Standard speech recognition technologies are used in this system. Continuous, mfcc mixture Gaussian, 3-state, and triphone-based hidden Markov models are used for acoustic-phonetic transcription. Word connectivity is written in a CFG (context-free grammar). The speech recognizer generates a word lattice as the bundle of recognition candidates. As new candidates are produced, the lattice grows frame by frame. The system sends the intermediate speech recognition results to a dialogue manager by using the ISTAR (Incremental Structure Transmitter And Receiver) protocol. This protocol enables to reconstruct the word lattice exactly in the dialogue manager [4][5].

2.2 Dialogue Management

As described in Section 1, speech recognition errors may cause fatal conflicts of the dialogue scene identification. The dialogue manager should be designed without state transition models.

First, we simplify the task semantics. The target of our system is to get appropriate slot values for task completion from user utterances through interactive dialogues. Quite a few practical applications result from such task-oriented

```
(1) $phrase -> $room kaigi-shitsu onegai-shimasu { slot=room, val=$room.val, prio=3 }
                    (meeting room) (please)
(2) $phrase -> $room kaigi-shitsu                 { slot=room, val=$room.val, prio=2 }
(3) $phrase -> $room                              { slot=room, val=$room.val, prio=1 }
(4) $room   -> dai-ichi (The first)               { val=1 }
(5) $room   -> dai-ni   (The second)              { val=2 }
(6) $room   -> dai-san  (The third)               { val=3 }
```

Fig. 2 An example of phrase-pattern detection rules
(Augmented Context-Free Grammar)

slot filling semantics. This simplification enables the emergent computational approach based on slot-filling phrase detections.

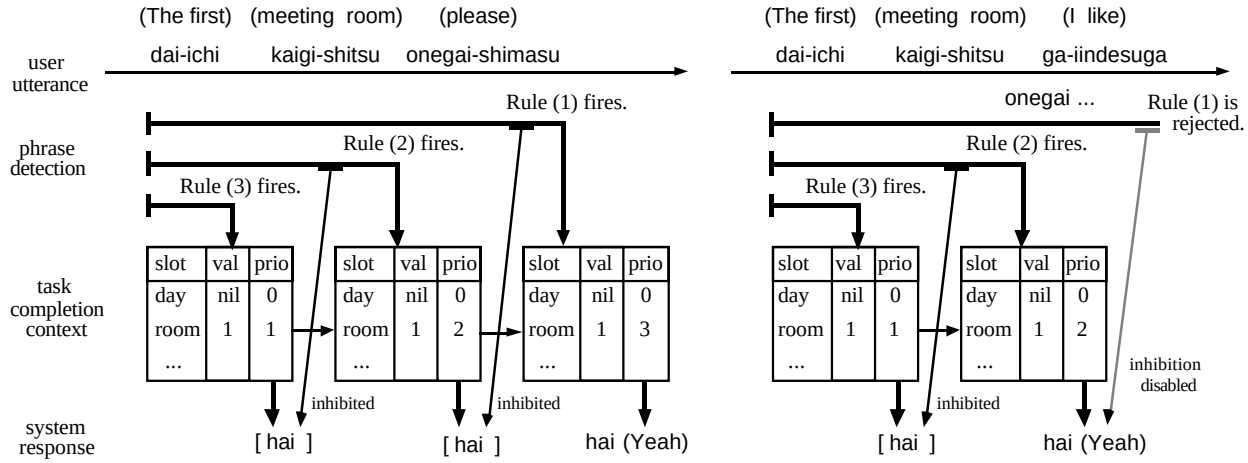
Second, we introduce the “task-completion context” defined as a set of semantic slot-values and their current priorities. The priority indicates the reliability of the slot value. The system updates the task-completion context according to incoming slot-filling signals and generates the next system response.

The input sensor of the ECL-DIALOG architecture is "phrase pattern" detection. Multiple CFG parsers search for slot-filling phrase patterns in user utterances. Many error phrases (false alarms) are detected, because the speech recognition is not accurate enough. For example, a user utterance "(I'd like to reserve on Monday)" is also recognized as "(yeah) (right) (Tuesday) (on Monday)", where parentheses bind each detected phrase. Our system generates a slot-filling signal for each detected phrase and attaches a priority to the signal according to its phrase pattern. For example, the idiomatic phrase pattern "(I'd like to reserve $obj on $day_of_week)" is given with a high priority. The simple word phrase "($day_of_week)" is given with a low priority because such a short phrase often appears at inappropriate places as a false alarm.

Another essential element of our architecture is the task-completion context. The task-completion context is a set of semantic slots that complete the task and the current priorities of the slots. The emergent phrase-pattern detectors incrementally generate numerous slot-filling signals. The system integrates them into the task-completion context and generates the next system response. For example, when the system finds a new slot value from user utterances, the system generates chiming utterances ("yeah", "muh", "hum hum"). When the slot signal has the same value as the context with a higher priority (than that of the context slot), the system silently updates the priority of the context slot. For example, when the context slot $dow="Monday" is replaced by a different value $dow="Tuesday" that has a lower priority, the system asks for confirmation of the slot change by sending "On Tuesday?".

An example of phrase-pattern detection rules written in an augmented context-free grammar is shown in Fig. 2. Slot-filling and priority-setting operations are specified in the augmented part braced by “{“ and “}”. The grammar rule (2) has higher priority than the rule (3), because it represents more complex and redundant phrase patterns than the rule (3). The rule (3) is subsumed by the rule (2). In the same way, the rule (2) is subsumed by the rule (1).

Figure 3 (a) shows an example of chiming response by the system. The user utterance is “Dai-ichi kaigi-shitsu onegai-shimasu (The first meeting room please)”. After the word “Dai-ichi”, the rule (3) fires and generates a slot-filling signal (slot=room, val=1, prio=1). This signal updates the task completion context and tries to generate a chiming utterance “hai (yeah)”. The system inhibits this response because the higher-priority rule (2) is currently active. After the word “kaigi-shitsu”, the rule (2) fires and generates a slot-filling signal (slot=room, val=1, prio=2). In this case, the signal has the same value as the context with a higher priority. The new



(a) example 1

(b) example 2

Fig. 3 Examples of system response generation (chiming)

priority overrides the context. After the words “onegai-shimasu”, the rule (1) fires and generates a slot-filling signal (slot=room, val=1, prio=3). Here, the system generates a chiming utterance “hai (yeah)”.

Figure 3 (b) shows another chiming example. The user utterance is “Dai-ichi kaigi-shitsu ga-iindesuga (The first meeting room, I like)”. This phrase pattern is not represented in the grammar of Fig. 2. The chiming utterance is generated when the rule (3) is rejected. The last context is (slot=room, val=1, prio=2).

Figure 4 shows an example of confirmation response by the system. The user utterance is “Dai-ichi kaigi-shitsu onegai-shimasu (The first meeting room please)”. The false alarm “dai-san (the third)” was detected and generates a slot-filling signal (slot=room, val=3, prio=1). Because the priority of this signal is lower than that of the current context, the system decides to confirm the new value to resolve the conflict.

2.3 Response Generation

All system guidances and responses are human’s pre-recorded voices. Variations of slot value representations are recorded for confirmation utterances. Especially, functional phrases are pronounced in prosodical contexts. The system concatenates them and generates system utterances with the intention. An acoustical echo canceller is used for avoiding the system to respond its own voices.

3. CONCLUSION

A new dialogue management architecture for human-machine speech communication was proposed. The emergent computational mechanism enables an incremental, non-deterministic and quick-response spoken dialogue system. Dialogue coordination and confirmation behaviors of the spoken dialogue system is useful to achieve high task completion performance.

ACKNOWLEDGMENT

The author wishes to thank Dr. Ken'ichiro Ishii, the head of the Information Science Lab. at NTT Basic Research Labs., and the DUG (Dialogue Understanding Group) members for useful suggestions.

REFERENCES

[1] Kobayashi, Y. et al : “Keyword Prediction in a Speech Dialog System,” Tech. Rep. JSAI, SIG-SLUD-9201-3, pp.19-26 (1992)

[2] Kawabata, T. : “Topic Focusing Mechanism for Speech Recognition based on Probabilistic Grammar and Topic Markov Model,” ICASSP-95, WA02-10, pp.317-320 (May 1995)

[3] Brooks, R. : “The Whole Iguana,” Robotics Science, pp.432-456, MIT Press (1989)

[4] Hirasawa, J. et al : “Implementation of Coordinative Nodding Behavior on Spoken Dialogue Systems,” ICSLP-98. (Dec. 1998)

[5] Gorz, G. et al : “Research on Architectures for Integrated Speech/Language Systems in Verbmobil,” COLING-96, pp.484-489. (1996)

[6] Hirose, K. et al : “A System for the Synthesis of High-Quality Speech from Texts on General weather Conditions,” Trans. IEICE, E76-A, 11, pp.1971-1980 (Nov. 1993)

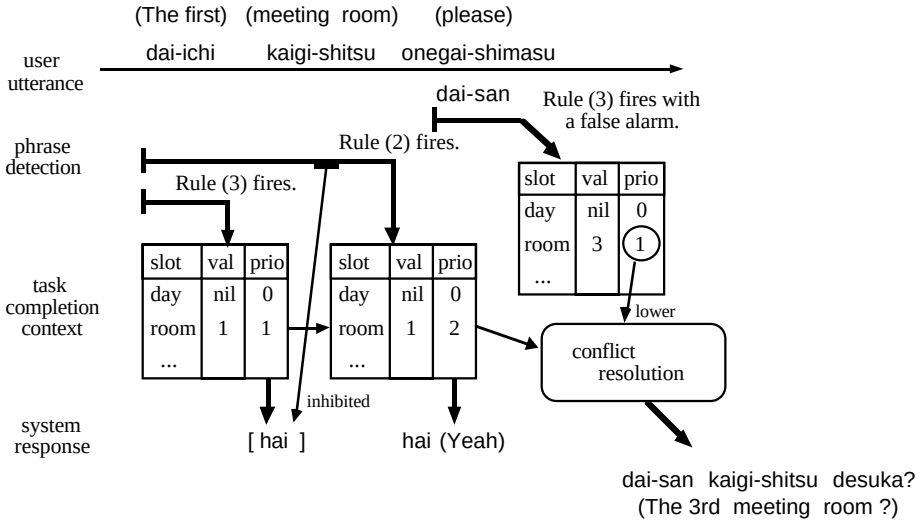


Fig. 4 An example of system response generation (confirmation)