A TASK-INDEPENDENT DIALOGUE CONTROLLER
BASED ON THE EXTENDED FRAME-DRIVEN METHOD

Yasuha NIIMI, Tomoki OKU, Takuya Nishimoto and Masahiro ARAKI

Kyoto Institute of Technology
Matsugasaki, Sakyo-ku, Kyoto 6068585, Japan
{niimi, nishi, araki}@d.j.kit.ac.jp, tomoki@vox.d.j.kit.ac.jp

ABSTRACT

This paper presents a task-independent dialogue control scheme, which is an extension of the frame-driven method. A spoken dialogue system, SDKIT-3, which is based on this scheme, has been developed for the tourist information service and has demonstrated it can manage problems which are encountered in a spoken dialogue system. The scheme is transportable to another task, the guide for personal computer users. Fifty dialogues were collected and used to design the topic frames for the new task and to test the new system. Simulated dialogues controlled by the newly created topic frames have proven the proposed scheme can be task-independent within a kind of tasks.

1. INTRODUCTION

The recent advance of speech technology has made it possible to build continuous speech recognition systems working in real time. Using such a system as an interface, we can construct a human-machine dialogue system which can complete a goal-oriented task through multi-utterance conversation. The dialogue management is an important component of such a system. Its role is to make interactions with a user as natural and efficient as possible while directing the dialogue not to stray from its domain and capability. The dialogue system needs to respond to user initiative and keep track of information and goal given by the user. It also needs to take the initiative to collect necessary information to attain the goal and make confirmations when it finds contradiction in information provided by the user.

There have been several approaches to dialogue management. An approach, which is commonly used, is to specify dialogues with a directed graph[1],[2]. In this paradigm each node of the directed graph is associated with a question the system can ask, as well as an action to be taken, and each edge is associated with user’s response to the question, which determines to which node the system should move next. The advantage of these systems based on this approach is that they are straightforward to design and their behavior is predictable. The disadvantage is that the dialogue tends to be inflexible and system-initiated.

In order to overcome this disadvantage, in other words, to allow the system to operate in the mixed initiative mode, the frame based dialogue control scheme has been proposed. The first version of this method was used in the text based dialogue system, GUS and demonstrated its ability to allow mixed initiative dialogues[3]. Spoken dialogue systems based on this paradigm have also been reported[4],[5].

In this paper we extend this idea to manage a larger task and to build a task-independent dialogue controller. It has been known that topics in a goal-oriented dialogue move according to a task-dependent tree structure[6]. This means that a goal-oriented dialogue could be divided into some subdialogues, each corresponding to one of subtasks. Employing this property of goal-oriented dialogues, we can structure a set of topic frames, given a task, which forms a tree structure by allowing slots to be filled with other frames, and develop a frame handler which uses this tree to control dialogues. Thus we have a task-independent dialogue controller which can control naturally both the shift of topics and the exchange of initiatives.

We will outline a spoken dialogue system we are developing based on the idea mentioned above and how to represent a dialogue history therein in section 2. Then we will describe how the frame handler can control dialogues in section 3, and an experiment which was carried out to test the portability of the dialogue controller in section 4. Lastly section 5 concludes the paper.

2. REPRESENTATION OF DISCOURSE HISTORY

2.1 Outline of SDKIT-3

We are developing a spoken dialogue system based on the extended frame-driven method, which is called SDKIT-3 (Spoken Dialogue System at Kyoto Institute of Technology). Its original task is information service for sightseeing in Kyoto. Fig.1 illustrates an example of the dialogue which the system would have with users. As shown in Fig. 1, SDKIT-3 works in both system initiative and user initiative modes, and
can understand a phrase input like U02, and an indirect response like U03.

S01 This is the tourist information service in Kyoto. Can I help you?
U01 I'd like to do a day tour in Kyoto.
S02 What are you interested in?
U03 Gardens.
S03 I see. Gardens in what a style would you like to visit?
U03 Gardens of Muromachi era.
S04 Among famous gardens in Muromachi era are the gardens of Ryoanji temple and Ginkakuji temple.
U04 The garden of Ryoanji is a famous stone garden, isn't it?
S05 That's right.

Figure 1: An example of the dialogue.

In spoken dialogue systems, the analysis of dialogue structures plays important roles in interpreting utterances. The discourse history can be described in terms of topics and dialogue acts[7]. SDSKIT-3 has two knowledge sources to describe the dialogue history. One is a transition network of dialogue acts which describes possible transitions from a dialogue act to another. The other is a set of topic frames which form related topics into a frame which might appear in sightseeing dialogues. Since slots of a frame can take other frame as their value, a set of topic frames forms a tree as whole, which we call a topic tree. We assume topics develop along this topic tree during a dialogue. This forms a subtree of the topic tree, which we call a dynamic topic tree. The topics in the dialogue illustrated in Fig. 1 are specialized along the structure as shown in Fig. 2.

Dialogue acts and topics are extracted through bottom-up and top-down analyses[8]. Bottom-up candidates for dialogue acts and topics are decided by applying a set of rules specially designed to the semantic interpretation of an utterance. Top-down candidates for dialogue acts and topics are decided by using the current state of the dialogue history which is described by a trace in the transition network on the dialogue act and the dynamic topic tree. Then, the logical ANDs between the bottom-up and top-down candidates are taken to decide the dialogue act and topic of an utterance of the user.

2.2 Topic Frames
A topic frame is a set of slots. A slot is described by a slot name, a value, a method for filling the slot, and an action of the controller after the slot has been filled. In addition, the slots of a topic frame are given the order of filling themselves. There are four ways to fill an unfilled slot: (1) to ask a user, (2) to retrieve instances from a database, (3) to use the default value attached to the slot, and (4) to link one of other topic frames.

The dialogue controller takes one of three actions after a slot has been filled; these are (1) to move the control to the slot with the next highest priority, (2) to move the control to the topic frame just linked, and (3) to present the information a user has asked.

SDSKIT-3 has three types of the topic frame: t-frame, p-frame and i-frame. T-frames specify relatively large topics like a tour plan which could be divided into subtopics. These subtopics are described by slots of t-frames. T-frames are expected to be located in the higher position on a hierarchy of subtopics in the dialogue. Fig. 3 illustrates an example of the t-frame, tour-plan-frame. A tour-plan-frame specifies a plan of a tour and has two slots, which are period-of-tour-slot and plan-slot. A plan-slot is filled with a pointer to a day-plan-frame. A day-plan-frame is used to describe a plan for a day tour.

```
{ tour-plan-frame
  [ name: period-of-tour
    value: [ ] : integer
    how-to-fill: ask-user
    action: move-to-next,
    name: plan
    value: [ ] : day-plan-frame
    how-to-fill: link-topic-frame
    action: move-to-frame
  ]
}
```

Figure 3: An example of the topic frame.

P-frames are used to describe database queries, and have formats similar to the record of a relational database. When some of its slots have been filled, a p-frame is converted to a database command, which retrieves instances satisfying the conditions described by the p-frame. An i-frame has almost the same format as a p-frame, and is used to store an instance retrieved from the database. It is connected to an instance slot of a p-frame, forming a child node of that p-frame.
2.3 Dynamic Topic Tree
As we reported in [9], a dynamic topic tree is not a simple tree, but an AND-OR tree. In the AND-OR tree, AND-nodes represent topics introduced by a user, and OR-nodes represent topics introduced by the system. If the user inquires about two or more sights (each assumed to be a topic), the system must offer information on all of them. On the other hand, even if the system proposes two or more candidates for a visit, the user is not interested in all of them, but might move to the other topics. An AND-OR tree is suited to reflect this difference. Furthermore, a topic frame has such slots that must be filled for the topic to be closed. These slots considered to be AND-successors of that topic frame.

Each topic frame of a dynamic topic tree has a flag to specify the state of it. A flag takes as its value one of 'current', 'unvisited', 'suspended', and 'closed'. 'Current' indicates a slot of that topic frame is focused. This means the subdialogue on that topic continues. 'Unvisited' indicates the topic frame has been introduced by either the user or the system, so being connected to the dynamic topic tree, but the subdialogue on that topic has not been opened. 'Suspended' means that a subdialogue on that topic frame has been opened, but is suspended by opening the subdialogue on a subtopic of that topic. 'Closed' means subdialogues on that topic have been closed.

3. TASK-INDEPENDENT
DIALOGUE CONTROL

3.1 Labeling procedure of the Dynamic
Topic Tree

Here we will explain how to examine whether a topic is closed or not. Leaf nodes of a dynamic topic tree are i-frames. They specify sightseeing spots, restaurants, hotels and so forth in our task domain. These topics are decided to be closed when they are presented to the user, and the user tells the system whether he includes them in his tour plan or not. When the user tells nothing about them, they are marked as 'suspended'. Thus the rules to close non-leaf nodes can be stated as follows.

1. A node with AND-successors can be closed only when all the successors have been closed.
2. A node with OR-successors can be closed when at least one of the successors has been closed.

We call these rules topic closing rules. The dialogue manager of SDSKIT-3 obeys strictly these rules to close topics. However, since we cannot expect the user to be so strict that he always shifts topics according to these rules, we have designed the dialogue manager to allow the user to shift topics in any way. Thus the flag of nodes is updated by the following rules.

1. When the topic moves from the current node to one of its child nodes, the current node is labeled as 'suspended' and that child node as 'current'.
2. When the topic moves from the current node to its parent node, the current node is closed and the parent node is labeled as 'current'.
3. All the child nodes of a closed node are closed.

It should be noted that the dialogue manager obeys the topic closing rules stated above when it moves the topic upward, but the user is allowed to violate them. After user and system utterances have been processed, the flags of all nodes of the dynamic topic tree are changed by applying these three rules recursively both upward and downward. We call this procedure a labeling procedure.

3.2 Frame-based Dialogue Control Scheme

As mentioned previously, SDSKIT-3 represents the dialogue history on the topic with a dynamic topic tree and retains a pointer to the focused slot of the node whose flag is 'current'. This is the situation that SDSKIT-3 is about to interpret user utterances or to speak to the user. Fundamentally the behaviors of SDSKIT-3 are subjected to the structure of the dynamic topic tree. However, it acts differently in interpreting user utterances and in speaking to the user.

First we will explain how it works in the interpretation of user utterances. As outlined in the previous section, the bottom-up discourse analysis produces candidates for the topic, and the top-down discourse analysis searches for some of the bottom-up candidates in the dynamic topic tree equidistantly from the current node. The distance between two nodes is defined by the number of edges which must be passed to move one node to another. For example, the distance between a node and its parent (or its child) is one, and the distance between two sibling nodes is two. When nodes without a child node are found in the search, they are expanded by copying corresponding subtrees of the topic tree.

Next we will explain how the dialogue manager works in speaking to the user. Its behaviors are quite simple in this case. It searches for an unfilled slot in the dynamic topic tree in the depth-first way from the current node, and tries to fill that slot. When the current node has been closed by the utterance the user just issued, the dialogue manager first applies the labeling procedure explained in the previous section to backtrack on the dynamic topic tree to the first unclosed node, and makes it the new current node. Then the depth-first search restarts from that node to find an unfilled slot. This is the principle of the topic shift by the dialogue manager.

4. PORTABILITY OF THE
DIALOGUE CONTROL SCHEME

The dialogue control scheme mentioned above has been implemented for the tourist information service and has demonstrated that it can manage several problems which are encountered in a spoken dialogue system [10]. In this section we will report a simulation experiment which was conducted to prove it could be
applied to other tasks than the tourist information service. We chose a guide for personal computer buyers as a new task. In this environment users who want to have a personal computer, conversing with the dialogue system on their purpose of having a computer, its desirable capacity and so on, can decide the specification of a computer.

For the simulation experiment we collected fifty dialogues on the new task through the WOZ method. In this dialogue collection the person who played a role of the wizard is different from the conductor of the experiment. Of fifty dialogues thirty were used to create the topic frames for the new task, and twenty were used to test the new system. Since the simulation experiment placed an emphasis upon testing the task-independency of the proposed dialogue control scheme, only the dialogue controller (abbreviated as DC below) was tested. For this purpose each utterance in the test corpus was manually labeled with a dialogue act, a topic and a focus.

The simulation experiment was conducted as follows. First DC is activated after the initialization. Its action after the self-introduction is to ask a user about the purpose to use a computer. Then the dialogue act, topic and focus assigned to the first utterance of the user are fed to DC. If, following the topic frames newly created, decides its next action, which is represented by a dialogue act, a topic and a focus corresponding to its next utterance. If the action of DC is the same as the one of the wizard, the labels of the next utterance of the user are fed to DC. Even if the question of DC happens to be different from the one of the wizard, it might be concerned with the specification of a computer and mostly asked elsewhere in the dialogue by the wizard. If this is the case, the answer to that question is picked up and input to DC. If the question of DC is not included in the dialogue, it is neglected and the next utterance of the user is fed to DC, or the experimenter makes an appropriate action. The former case simulates a kind of indirect responses by the user.

Since the designer of the topic frames is different from the wizard in collecting the dialogues, a simulated dialogue is generally different from the original dialogue in the corpus. We, therefore, evaluated simulated dialogues by whether they attained the same goals as the ones of the original dialogues. In each of the simulated dialogues we were able to have the same specification of a computer as the one attained in the corresponding WOZ-dialogue.

We conducted one more preliminary experiment similar to the one explained above, on the task domain of hotel reservation and assured dialogues including simple negotiation on the room charge could proceed as expected. These two experiments prove the proposed dialogue control scheme can be task-independent within a kind of tasks.

5. CONCLUSION

This paper has presented a task-independent dialogue control scheme in a spoken dialogue system, SSDKIT-3. It is an extension of the frame driven method. In SSDKIT-3 the discourse history on the topic is represented with a dynamic topic tree whose nodes are topic frames. In the interpretation of user utterances, SSDKIT-3 searches for the topic of the user in the dynamic topic tree equidistantly from the focused node. In speaking to the user, it searches for an unfilled slot in the dynamic topic tree in the depth-first way from the focused node, and tries to fill that slot, for example. Although this principle is quite simple, two experiments demonstrated proposed scheme could control dialogues flexibly and be task-independent within a kind of tasks.

References