Understanding and Presentation of Conceptual Information based on Planning

Kuniaki UEHARA and Minoru TANIGUCHI
Department of Computer Science and Systems Engineering
Kobe University
Nada, Kobe 657, JAPAN
e-mail: uehara@kobe-u.ac.jp

1 Introduction

Almost all current dialogue systems are generalization-based, in the sense that they use explicit abstract generalizations. Within computational linguistics, two main approaches have dominated, the first using schemata to constrain the overall organization of the text [4], and the second using plan operators to generate a sequence of utterances given particular communicative goals (i.e. traditional planning) [2][5].

Schema-based approaches cannot participate in an interactive dialogue with users. In particular, they cannot elaborate on previous utterances or respond to follow-up questions in the context of the on-going dialogue. The problem is that they fail to provide an explanatory account of why and how a particular schema is structured in a particular way, and as a result are not easy to adapt to a particular discourse. This means that they view generating responses as a one-shot process. That is, a system is assumed to have only one opportunity to produce a response that the user will find satisfactory.

In order to produce a suitable context for follow-up questions, schema-based approach requires an enormous amount of detailed knowledge about the user, i.e., user model. However, it will be impossible to build such complete and correct user models. An alternative requirement is that they must contain a schema that precisely matches the dialogue being processed. Furthermore, expectations represented in a schema must be at a very specific level. But this means that new dialogues must first be analyzed in detail to determine the appropriate schema. In short, schema-based approach is stymied when confronted with situations that stray even slightly from its specific expectations.

Plan-based approaches are limited because they know nothing of stereotypes, and they treat each dialogue as completely novel. This limitation dooms them to perform expensive text planning every time they recognize a new instance of dialogue. Furthermore, researchers who take this approach must concentrate on methods of controlling the combinatorial explosion inherent in the text planning process. Finally, since the form in which knowledge is available from people is different from the form in which knowledge is represented in dialogue systems (i.e. representation mismatch), it is difficult to transfer the "deep" text planning knowledge from human to machine, and, thus, the text planning knowledge in a dialogue system does not suffice to respond to follow-up questions.

In this paper, we will propose a case-based text planning approach to dialogue systems in which feedback from the user is an integral part of the text planning process. Case-based text planning approach has the following advantages: (1) text planning process can start immediately after a case base had been constructed, (2) as discourse situation changes, the dialogue system can re-plan responses reactively, which allows the system to accept various follow-up questions from the user, by revising the current plan suggested by the retrieved case, (3) computational cost will be reduced since sub-optimal solutions can be retrieved without performing complex text planning which must be done in a traditional planning approach, (4) the system may use information in a user model if it exists, but not require it.

2 Case-Based Text Planning

2.1 Basic Ideas

Case-based planning is the paradigm that devises new plans by retrieving and adapting old ones from memory. Following Larry Birnbaum [1], an explicit model of case-based planning is as follows: Problem anticipation is the process of predicting problems in planning for a given goal. Plan retrieval is the process of searching for a plan that satisfies the goal while avoiding problems. Adaptation is the process of modifying the plan to satisfy the goal. The basic model of case-based planning is obviously incomplete if we adopt the model as the text planner of a dialogue system. It is incomplete because this mechanism ignores the following problems:

1. Users frequently ask follow-up questions requesting clarification, elaboration, or re-explanation.
2. Users may misunderstand the explanation if they don't have enough or accurate knowledge about the domain.

In order to deal with the former problem, we can make use of a reactive approach which employs feedback from the user to guide subsequent dialogues. To accomplish this task, the system must not only be able to generate and execute text plans, but must be willing to interrupt and modify them, when the user asks a follow-up question or indicates that he does not understand the explanation.

A plausible way to deal with the latter problem is to assume that it will be possible to build complete and correct user models, although this assumption is questionable. By throwing out the user model assumption, we will adopt an alternative approach where the system learns to anticipate and avoid user's misunderstanding that it has previously encountered. When the misunderstanding re-occur in later situations, the text planner is reminding of the past misunderstanding and this reminding serves as a warning to the text planner that it has to plan for the fact that this misunderstanding is going to occur again. The text planner also stores the repaired plans that were built in response to past misunderstanding.

2.2 Case-Based Text Planning

The system we use for controlling and carrying out the reactive dialogue system is called case-based text planner. If the communicative goal is extracted from the user's utterance, the goal is sent to the case-based text planner. The planner retrieves the previous problem case that has the similar potential for the same problem to achieve the communicative goal. After predicting the problem, the planner retrieves the plan case that satisfies the communicative goal as much as possible. If not, the planner tries to build up the new plan from scratch by applying a base of plan operators. The constructed plan is treated as a tentative plan case by the planner.

Generally, both the retrieved problem case and plan case do not fit in with the current communicative goal completely. Therefore, they must be adapted to satisfy the goal. After adaptation, problem case and plan case are integrated into a new plan case. With a plan case of this kind of integration, the planner is able both to prevent the previous problem as a possibility for the current situation and to provide the solution to achieve the user's communicative goal.

At this moment, the integrated plan is ideal in the sense that it is a fixed, static plan for a highly predictable situation in which the user's utterance can be anticipated completely. However, in a real world domain, user's utterance cannot be anticipated, the dialogue cannot be controlled, and detailed plan cannot be built in advance. In order to avoid this deficiency, the plan is executed by observing the user's feedback a step at a time. If the feedback is different from the one that is expected by the planner, the planner modifies the portion of the plan to produce the desired response. This modified text plan is also recorded as the plan case for future use.

3 Case Representation

3.1 Plan Case

Plan case is represented as a network, in which each node corresponds to the system's atomic communicative action . Communicative action is denoted as a predicate whose arguments specifies the utterance situation. Communicative action can be realized as a variety of surface form. Each node in a network is connected by arcs which are labeled with the user's feedback. User's feedback is also characterized by the kind of information it conveys. In addition, an arc labeled with grounds specifies the domain knowledge from which the system's communicative action is derived. The domain knowledge plays a part in constraining the adaptation of the plan case.

The plan case is indexed by the initial goal which is one of the user's feedback. The planner uses the conceptual hierarchy about the domain to determine the similarity of the user's goal and initial goal in the plan case. The similarity metric used in our implementation is called knowledge-based pattern matching. Fig. 1 depicts the simplified version of a plan case.

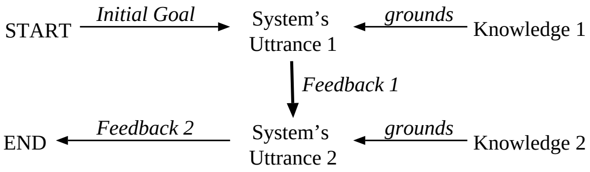


Figure 1: The simplified plan case

3.2 Problem Case

Studying advisory consultation between humans, we found that advice-seekers frequently commit an error because of their careless mistake, missing knowledge, mistaken knowledge, or oblivion. These errors are stored as a problem case. The problem case serves as a warning to the planner that it has to plan for the fact that this problem is going to occur again. Problem cases are represented as frame or slot-and-filler notation. The problem case is indexed by the content of situation slot. The planner also uses knowledge-based pattern matching to determine the similarity of the user's goal and the value of situation in the problem case.

Problem	description of the user’s problem
Situation	contextual situation where the failure was occurred
Solution	solution to recover the failure
Prevention pr	evention to avoid the failure

Figure 2: Problem case representation

4 Answering Questions

Users frequently do not state their problem directly. Instead, they may expect the system to infer their problem from their utterance. Thus, if the user asks a question or indicates that he may have the problem, the planner analyses the question and extracts the user’s problem by using heuristics shown in Fig. 3. After finding the user’s problem, the planner searches for both the problem case and plan case so as to fix the user’s problem.

Oblivion:
goal(Situation), user_knowledge(Situation) -> forget(Situation).
Careless mistake:
not(intend(Situation)), fact(Situation) -> careless_mistake(Situation).
Insufficient understanding:
executed(Action, fail), recover_to_fail(Situation) ->
insufficient_understanding(Situation).
Incorrect understanding:
executed(Action, succeed), recover_to_fail(Situation) ->
incorrect_understanding(Situation).

Figure 3: Heuristics to find out the user’s problem

If the user interrupts in the middle of the explanation with some feedback, the planner must be able to respond to the feedback, yet resume the previous interrupted dialogue plan in a clear way so that the explanation may be completed. Fig. 4 enumerates and exemplifies the user’s feedback, although some of them were borrowed from [5].

understanding	”Uh huh”, ”I see”
affirmation	”I found it.”, ”I’ve done it”
repetition	” So x, y, and z caused the problem.”
negation	”No”
answering(X)	”That is X”
elaboration(X)	”What is X?”
confusion	”Huh?”, ”What do you mean?”
causal	”How?”, ”Which causes what?”
rejection	”Shut up”, ”Quiet!”
temporal	”And then?”, ”Next?”
interruption	”Incidentally”, ”Did you hear about?”
goal(X)	”I want to perform X”

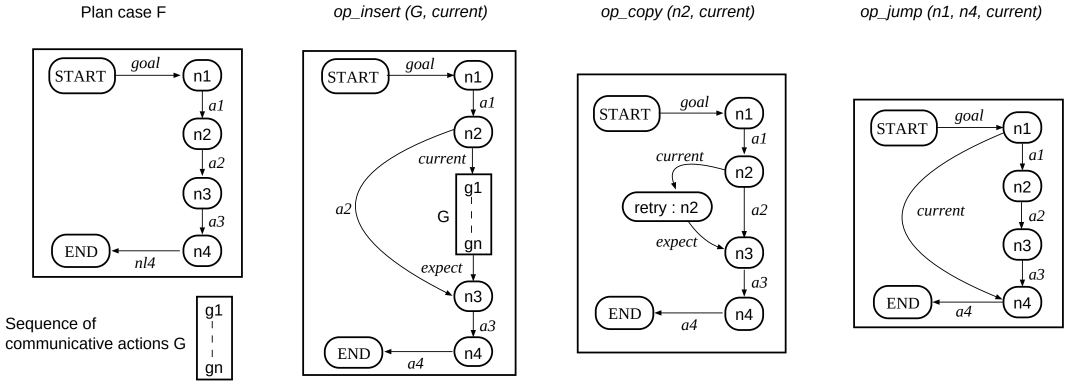
Figure 4: User’s feedback [5]

User's feedback guides the selection of appropriate tweak rules during the modification. Tweak rules given in Fig. 5 are associated with the user's feedback shown in Fig. 4 respectively.

Feedback	Tweak rule
understanding	do_nothing
affirmation	do_nothing
summary	op_insert(agreement, current)
elaboration(X)	op_insert(plan(know_con(X)), current) & op_copy(previous, next)
confusion	op_insert(alternative, current)
rejection	op_jump(current, end)
temporal	do_nothing
interruption	op_branch(plan(NewGoal)) or op_insert(rejection, current)

Figure 5: Tweak rules

Tweak rule consists of a set of atomic transformation operators. Fig. 6 shows a set of some atomic transformation operators. For example, general insertion operator inserts a sequence of communicative actions into the current plan case. Subgoal-preserving substitution operator substitutes an communicative action in the current plan case by another communicative action (or sequence of communicative actions) that produce the same effect. This operator is useful if the user's feedback in the original plan case cannot be satisfied, or if the presence of a particular communicative action violates the user's goal.



(a) The current plan case (b) General insertion (c) Subgoal-preserving substitution (d) General deletion

Figure 6: Atomic transformation operators

5 Conclusion

This paper describes an approach to reactive text planning based on the idea of case-based planning. The system described here were implemented as a natural language consultation system for the UNIX operating system, called ASSIST-R. Users can ask ASSIST-R how to do things in UNIX, get definitions of UNIX terminology, and get help debugging problems in using commands. Currently, we are using an adaptation of Dyer's McDYPAR [3] for the Japanese query analyzer and Japanese Tree Adjoining Grammar for text generation mechanism [6]; however, since the focus of our research is text planning, we make no claims about parsing and text generation.

References

[1] Birnbaum, L. and Collins, G.: A Model-Based Approach to the Construction of Adaptive Case-Based Planning Systems, Proc. of 1991 DARPA Case-Based Reasoning Workshop, pp.215-224 (1991).

[2] Cawsey, A.: Planning Interactive Explanations, Int. J. Man-Machine Studies, Vol.38, pp.169-199 (1993).

[3] Dyer, M.: In-Depth Understanding: A Computer Model of Integrated Processing for Narrative Comprehension, MIT Press (1983).

[4] McKeown, K. R.: Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text, Cambridge University Press (1985).

[5] Maybury, M. T.: Communicative Acts for Explanation Generation, Int. J. Man-Machine Studies, Vol.37, pp.135-172 (1992).

[6] Uehara, K.: Japanese Tree Adjoining Grammar and its Application to On-Line Help System NeoAssist, Proc. of the 1st International Workshop on Tree Adjoining Grammars, pp.88-100 (1990).

[7] Uehara, K. and Ogino, H: A Case-Based Approach to Text Planning, Proc. of the IASTED International Conference on Artificial Intelligence, Expert Systems and Neural Networks, pp.123-127 (1994).

Key Words: Dialogue system, case-based planning, text planning