SIS:A SHELL FOR INTERVIEW SYSTEMS

Atsuo KAWAGUCHI, Riichiro MIZOGUCHI,
Takahira YAMAGUCH1 and Osamu KAKUSHO

The Institute of Scientific and Industrial Research, Osaka University, 8-1 Mihogaoka, Ibaraki, Osaka 567, Japan

Abstract

An interviewer has two kinds of knowledge. One is about a domain under consideration and the other is knowledge for interview itself which makes the interviewer an expert of interview. The latter seems to be independent of the domain because an experienced interviewer, such as a TV interviewer, can carry on his tasks in many fields. Furthermore, we believe that the interview knowledge consists of several interview knowledge primitives. Based on this idea, we are developing SIS, a Shell for Interview Systems which has seven question strategy primitives as the knowledge for interviewing. Generality and effectiveness of SIS are shown through two implementation examples of interview systems, I^2s and MORE. The seven primitives are shown to be efficient for the two system whose domains and tasks are entirely different.

1• Introduction

Interviewing plays a crucial role in many fields to make ambiguous knowledge clear. In building expert systems, for instance, the heuristics are generally obtained through interview of a knowledge engineer with a domain expert. In software specification, interview is indispensable to extraction of requirements from customers.

The authors have been investigating interview as a discourse aiming at transferring (ambiguous) knowledge. The purpose of this research is to uncover the knowledge and mechanism necessary for interview, to discuss the question strategies used in interview and to develop a framework for interview systems. The authors have developed I^2S (an Intelligent Interview System) whose task is to interview a data base customer and to design logical structure of the data base as a part of this study so far[ll.

An experienced knowledge engineer can interview domain experts and construct expert systems in various domains. A TV interviewer can also interview persons in many fields. These observation suggest that they have domain independent skills (knowledge) for interviewing. In the development of I^2S, it was an important point to distinguish the domain-independent interview knowledge for logical design of data base from domain-dependent one and four domain-independent question strategies were obtained.

Moreover, such domain-independent knowledge for interviewing dependent on the task consists of several primitives independent of the task. For example, a careful comparison between the four question strategies of I^2S and those of a knowledge acquisition system, M0RE[2}, reveals some knowledge common to the both. This allows us to consider an interviewer as an expert system and suggests the possibility of a general framework for interview systems. This paper describes a Shell for Interview Systems, SIS for short, based on this idea. Of course, there are many other aspects of interview systems and knowledge acquisition systems such as R0GET[3], ETS[4], etc. The authors are especially interested in discussion on interviewer's question strategies inherent in interviewing.

Various interview systems can be constructed by using SIS with description of the meta knowledge of the domain and task. SIS has seven question strategy primitives. One can define various question strategies (dependent on the task) by specifying the combination of the primitives.

2. SIS: Shell for Interview Systems

SIS is a shell for constructing interview systems. A general flow of interviewing consists of three stages. First, an interviewer requires an interviewee of initial information which becomes a cue for the following interview. The interviewer constructs an incomplete domain model in this stage. And he usually has many questions at this time. We call such questions 'attentions'. In the second stage, the interviewer asks the interviewee additional questions according to the attentions. Finally the interviewer produces an output of the interview.

In order to carry out interviewing, an interview system must have the following functions and knowledge for them:

(a) to construct a domain model from information extracted from an interviewee,
(b) to make attentions in the first and second stages,
(c) to infer about the attentions and to ask the interviewee questions based on them,
(d) to make an output from the domain model and
(e) to analyze and generate sentences.

On the basis of the above discussions, we have designed an architecture of SIS. It consists of four modules such as parser, generator, memory and inference engine. The parser analyzes input sentences, while the generator generates sentences for the interviewee. They employ a demon-based mechanism. The memory contains knowledge for interviewing described by a system developer and information obtained through interview. Seven modules for question strategy primitives (QPs) built in the memory. They check the information obtained from the interviewee and generate attentions. The inference engine is essentially a

production system and processes the attention.

Domain model in SIS is a kind of network composed of domain_constituents (DCs) as nodes and domain_links (DLs) as arcs. DCs are represented in terms of frames, while knowledge for inference about attention is represented by rules in SIS, An interview system developer can describe the knowledge (a), (d) and (e) discussed above by defining prototypes of these frames with associated demons for parsing and generating, and knowledge (c) by defining rules with some extra information for controlling the system.

Whenever any information is given to the memory, it scans the domain model to check if any part of the model satisfies the conditions for making attentions. The conditions correspond to the knowledge (c). They are defined by specifying the objects of application of QPs,

QPs have their own conditions when to make attentions. They apply the conditions to all objects defined by the system developer and make attentions when the conditions are satisfied. The attentions indicate important issue underlying in domain model to be discussed with the interviewee. By defining inference rules for the attentions, the system developer can control the behavior of the system to the questions.

The detailed description of the QPs' conditions are shown below.

(1) planner (see Figure 1)
For paths and DCs obtained by tracing DLs from a DC,
(a) there are prototypes which could be chained to the paths,
(b) there are more than one path to reach one DC or
(c) there are paths from the same DC to different one.
The chainability in (1) is determined from intersection of predefined slot sets of each frame.

(2) script_tracer
There is a script for asking questions in a prototype corresponding to a DC.

(3) unknown_object_detector
Information which is not defined in prototypes are entered,

(4) discord__detector
Slot values of the DCs corresponding to each other are not the same,

(5) ambiguity__detector
There are more than one value in a slot,

(6) certainty_manager
A certainty factor assigned to a DL is out of the range the system developer defined.

(7) constraint__checker
A DCs slot value violates its constraint.

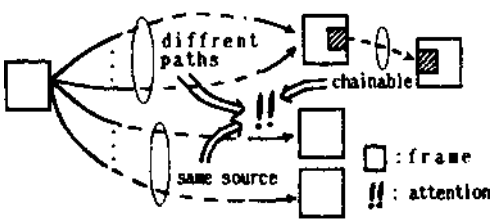


Fig. 1 Planner's conditions.

QPs can be used to detect many patterns which suggest the necessity of asking questions for more information.

3• Examples

This section describes implementation of two interview systems, which are entirely different in their tasks and domains. Effectiveness and generality of QPs will be shown through these examples.

3.1 I²S[1]

I²S (Intelligent Interview System for logical design of data base) interviews a data base customer and designs logical structure of data base according to the following steps:

(1) It asks questions of the customer (interviewee) to get requirements of a target data base as initial information,
(2) It constructs a domain model called plan-structure and refines it by asking the customer some questions,
(3) It makes conceptual schema of the target data base from the plan-structure as its output.

I²S employs the concept of plan to represent a domain. In the domain model called plan-structure, each noun or verb is represented as a frame. A noun frame contains several attributes which the corresponding entity has, while a verb frame has description of its activity represented in terms of the concept of plan.

Various question strategies are used by I²S to refine a plan-structure and design conceptual schema. We have obtained four kinds of strategies as follows:

(1) planning,
(2) making the detailed level of description equal,
(3) using scripts for nouns and
(4) interrupting when an interviewee mentioned 'time'.

While all of these strategies can be described in terms of QPs, we will show an example concerning planning strategy.

I²S reasons about activities to get unknown activities or relations between them based on the current plan-structure and shows the interviewee the results. We call this process planning. Suggestions presented by this planning strategy are very effective because it is rather difficult for a customer to prepare all data items necessary for his data base in advance. For instance, suppose a customer said a fact shown in Figure 2(a). A plan-structure corresponding to this fact is shown in (b). If a dictionary of I²S contains semantic description of 'stock' as Figure 2(c), I²S finds it and asks an interviewee some questions as shown in (d) because the precondition of 'construct' can be satisfied by the goal of 'stock'

In order to implement this strategy by SIS, a system developer defines prototype frames of verbs with appropriate demons for translating English sentences into DCs and vice versa. Then he implements the planning strategy by using planner's condition (1) to search candidate verbs. 'Found_candidate' attention will be generated when one of the following conditions are satisfied:

(1) there are some prototypes (DCs) whose goals are equivalent to precondition of

a DC (prototype).

(2) there are some DCs whose goals are equivalent to precondition of a DC,

And finally, he describes the knowledge of how to process the attention for planning strategy in rule form. After defining all of these strategies, SIS will automatically parse initial information into plan-structure, apply QPs, make attention, infer about them, and make question and answer with the interviewee.

The system works in an interview as shown in Figure 3. Suppose an interviewee mentioned the fact shown in Figure 2(a) as a requirement. First, the parser makes constituents of plan-structure from interviewee's input (1). Then, they are stored in the memory. QPs are applied to the constituents and in this example, the 'construct' DC and 'stock* prototype satisfy the condition for making found_candidate attention. The attention is stored in the attention list.

After obtaining the initial information, the system begins to process the attentions. Attentions are taken out from the attention list one by one (4) and sent to the inference engine. In this example, the system asks the interviewee questions as shown in Figure 2(a).

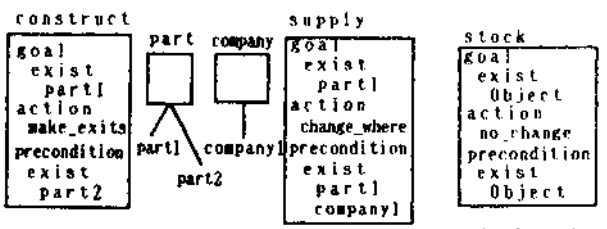
3.2 M0RE[2]

MORE is a knowledge acquisition system for constructing diagnostic expert systems. It first asks a domain expert about hypothesis, symptoms and other conditions (as initial information), then constructs a domain model. A domain model constructed by MORE is a network in which hypotheses, symptoms and conditions are represented as nodes and linked together by 'links' or 'paths'. Next, MORE applies eight question strategies to the model, asks the expert and refines it. Finally, it makes a rule set for the expert system from the model.

Among the eight question strategies, 'differentiation' is one of the most basic ones. A symptom S is said to differentiate hypothesis HI from H2 when there is a path from H1 to S, and no path from H2 to S. When there is a pair of hypotheses for which there is no differentiating symptom, MORE asks a domain expert about it.

Some parts are constructed from other parts
which are supplied from other companyies.

(a) Fact.



(b) Plan structure. (c) 'stock'.

I²S: I think you should store the information about 'stock'.
Do you agree ?
Customer: Yes.
I²S: Please enter an information requirement about 'stock'.
⋮

(d) Q & A.

Fig.2 Planning strategy.

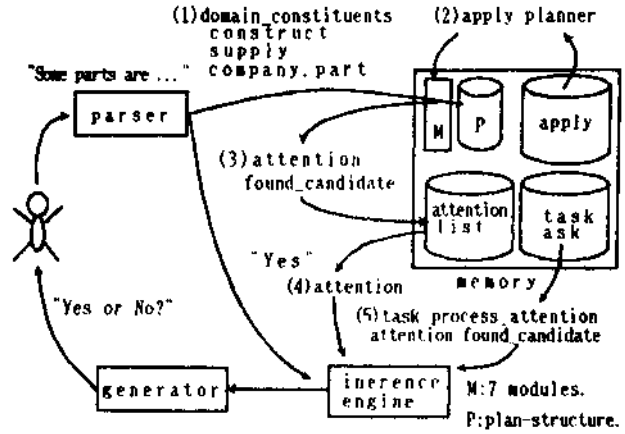


Fig.3 Processing example of planning stragety.

'Differentiation' strategy can be implemented in SIS by using planner's condition (c) to detect not-differentiated hypotheses. The planner is directed to start tracing from a symptom and to find paths from a symptom to different hypotheses. A domain-model will be automatically constructed by defining prototype frames of symptoms and hypotheses with demons for translating input information into frames linked each other. A system developer also describes rules for checking if there are really not differentiating symptoms when 'same_source' attention is made, and asking an expert new symptoms.

In an interview, when the pair of hypotheses to which there are same paths from a symptom are found, the attention 'same_source• will be generated. Then the system will infer how to treat the attention using the rules described by the system developer and ask an expert for a new additional symptoms.

4.. Conclusion

We have described SIS, a shell for interview systems. SIS has seven question strategy primitives for representing domain-dependent knowledge for interviewing. They are essentially detectors of problems in the network representing domain model and can be used as drivers for interviewing. The generality and effectiveness of the primitives have been evaluated through the implementation examples of two interview systems, I^2S and MORE.

SIS is under implementation on Hewlett-Packard's HP-9000 model 320 using C_Prolog.

References:

[1] Kawaguchi, A., et al., "An Intelligent Interview System for Conceptual Design of Database," Proc. ECAI'86, 1986.

[2] Kahn, G., Nowlan, S. and McDermott, J., "Strategies for Knowledge Acquisition," IEEE PAMI, Vol.PAMI-7, No. 5, 1985.

[3] Bennett, J. S., 'ROGET: A Knowledge-Based System for Acquiring the Conceptual Structure Of a Diagnostic Expert System," Journal of Automated Reasoning, 1, 1985.

[4] Boose, J. H., Expertise Transfer for Expert System Design. Elsevier, 1986.