

RESPONDING TO "HUH?": ANSWERING VAGUELY ARTICULATED FOLLOW-UP QUESTIONS

Johanna D. Moore

University of California, Los Angeles
and
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695

ABSTRACT

Expert and advice-giving systems produce complex multi-sentential responses to user's queries. Results from analyses of novice/expert dialogues indicate that novices often do not understand an expert's response and rarely ask a well-formulated follow-up question. Thus systems must be able to provide further information in response to vaguely articulated questions. However, current systems cannot clarify misunderstood explanations or elaborate on previous explanations. In this paper we describe an approach to explanation generation that expands a system's explanatory capabilities and enables the production of clarifying or elaborating explanations in response to follow-up questions or indication that the explanation was not understood.

KEYWORDS: Question-Answering Systems, Discourse Analysis, Text Generation

INTRODUCTION

Expert and advice-giving systems are expected to provide solutions and advice to users faced with real problems in complex domains. Systems that interact with users in these situations are required to produce complex multi-sentential responses, such as definitions of terms, justification of results, and comparisons of alternate methods for solving problems.

As responses become more complex we must face the problem that users may not understand the explanation produced and will need to ask follow-up questions. In a study of a "naturally occurring" expert system, Pollack *et al* found that user-expert dialogues are best viewed as a negotiation process in which the user and expert negotiate the statement of the problem to be solved as well as a solution the user can understand and accept [10].

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

In our own work on explanation, we too have examined samples of naturally occurring dialogues from several different sources: tape-recordings of office-hour interactions between first year computer science students and teaching assistants, protocols of programmers interacting with a mock program enhancement advisor, and transcripts of electronic dialogues between system users and operators [11].

From our analysis of this data, we have made several observations, among them:

- *Users frequently do not fully understand the expert's response.* Users rarely stated a problem, received a result or explanation, and then left, satisfied that they understood and accepted the expert's explanation. The expert frequently found the need to define terms or establish background information in response to feedback that the listener did not completely understand the response.
- *Users frequently ask follow-up questions.* Users frequently requested clarification, elaboration, or re-explanation of the expert's response.
- *Users often don't know what they don't understand.* Users frequently could not articulate a clear follow-up question. In many cases, the follow-up was vaguely articulated in the form of mumbling, hesitation, repeating the last few words of the expert's response, or simply stating "I don't understand." Often the expert does not have much to go on, but must still provide further information.

None of the currently existing explanation facilities are capable of clarifying misunderstood explanations or elaborating on previous explanations. In part, the explanation components of current expert systems are limited because they make use of simple techniques such as canned text or templates. However, even the more sophisticated generation techniques employed in computational linguistics are inadequate for responding to follow-up questions. The problem is that both expert system explanation and natural language generation systems view generating responses as a one-shot process. That is, a system is assumed to have one opportunity to produce a response that the user will find satisfactory.

However, our own and other analyses of naturally occurring data indicate that this one-shot model of explanation is inappropriate [10, 3, 14]. Moreover, if a system has only one opportunity to produce a text that achieves the speaker's goals without over- or under-informing, boring or confusing the listener then that system must have an enormous amount of detailed knowledge about the listener. This has led to a view that improvements in explanation will come from improvements in the user model and considerable effort has been expended in representing a *detailed* model of the user – including the user's goals, what the user knows about the domain, how information should be presented to that user, and so forth [1, 5, 9]. However, it is unlikely that we will be able to construct, verify, or effectively utilize complete and correct hearer models [13]. Further, by focusing on user models, researchers have ignored the rich source of guidance that people use in producing explanations, namely feedback from the listener.

There is a real disparity between what the data reveals, on the one hand, and the explanation facilities that current systems provide and the assumptions they make about how users interact with experts, on the other. We believe that a *reactive* approach to explanation is required – one in which feedback from the user is an integral part of the explanation process. A reactive explanation facility should include the ability to:

- monitor the effects of its utterances on the hearer.
- recover if feedback indicates the listener is not satisfied with the response.
- answer follow-up questions taking into account previous explanations – *not* as independent questions.
- offer further explanations even if the user does not ask a well-formulated follow-up question.
- use information in a user model if it exists, but not require it.

In this paper, we first discuss the limitations of current systems. We then present our approach to alleviating some of these limitations. We discuss the architecture of our explainer and present an example which demonstrates how our approach is used to offer an elaboration in response to the user's vaguely articulated indication that the explanation was not understood.

LIMITATIONS OF CURRENT SYSTEMS

There are two main reasons why current systems cannot deal with follow-up questions in a meaningful way. First, to be able to clarify a misunderstood explanation or respond to a follow-up question in context, a speaker must understand the explanation he has produced. Unfortunately, current expert systems produce explanations by filling in templates or using canned text and thus have little or no "understanding" of their own explanations. They do not represent the goal of the explanation, what rhetorical purposes are served

by individual clauses in the text, or what assumptions about the listeners knowledge may have been made. Thus, they have no means for diagnosing or recovering from a misunderstanding or for answering a follow-up question in the context of a previous response.

Second, making oneself understood often requires the ability to present the same information in multiple ways or to provide different information to illustrate the same point. However, current systems typically have only a single response strategy associated with each question type. Without multiple explanation strategies for responding to a single question, a system cannot recover from a misunderstanding even if it understands the cause.

We have built an explanation component for an expert system which addresses these problems. To provide the capabilities described above, we:

- plan responses such that the intentional structure of the responses is explicit and can be reasoned about,
- keep track of conversational context by remembering not only what the user asks, but also the planning process that led to the system's response,
- taxonomize the types of (follow-up) questions that are asked and understand their relationship to the current context, and
- provide flexible explanation strategies with many and varied plans for achieving a given discourse goal.

SYSTEM DESCRIPTION

Our explanation generation facility, which is the focus of this paper, is part of the Explainable Expert Systems (EES) framework [8]. When an expert system is built in EES, an extensive development history is created that records the goal structure and design decisions behind the expert system. This structure is available for use by the explanation facility.

We have used EES to construct a prototype expert system, called the Program Enhancement Advisor (PEA) [8], which we are using as a testbed for our work on explanation generation. PEA is an advice-giving system intended to aid users in improving their Common Lisp programs by recommending transformations that enhance the user's code¹. The user supplies PEA with the program to be enhanced. PEA begins the dialogue with the user by asking what characteristics of the program he would like to improve. The user may choose to enhance any combination of readability, maintainability, and efficiency. PEA then recommends transformations that would enhance the program along the chosen dimensions. After each recommendation is made, the user is free to ask questions about the recommendation.

¹PEA recommends transformations that improve the "style" of the user's code. It does not attempt to understand the content of the user's program.

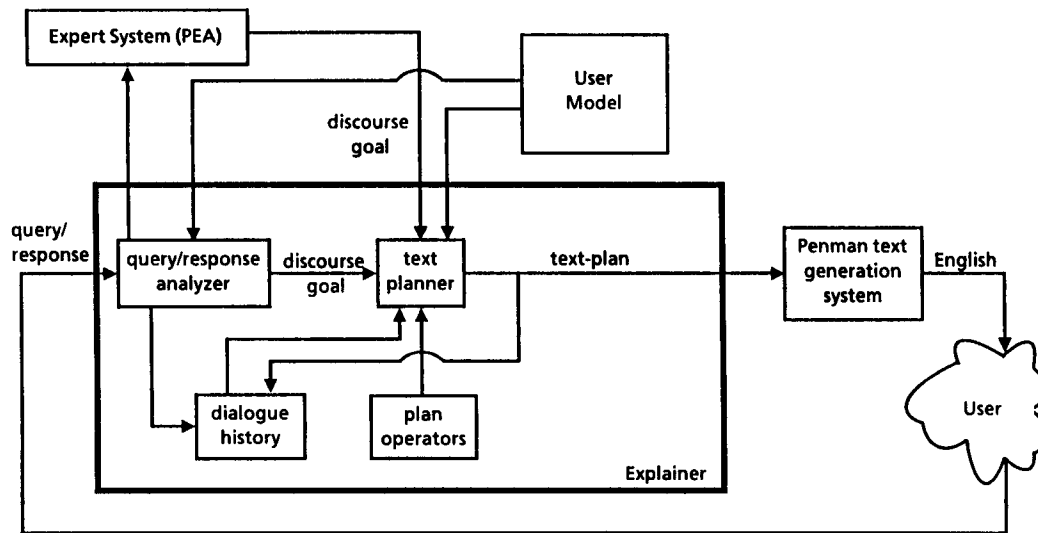


Figure 1: Architecture of Explanation System

An overview of the explanation generation facility (and its relation to the PEA expert system) is shown in Figure 1. To interact with the user, the expert system posts a discourse goal (e.g. persuade the hearer to do an act, describe an entity) to the *text planner*. A discourse goal may be posted as a result of reasoning in the expert system or as a result of a query from the user (see Figure 1). User queries must first be interpreted by the *query analyzer*. Even though we assume the user poses queries in a stylized notation,² ambiguities may still arise. It is the responsibility of the query analyzer to disambiguate the query and form the appropriate discourse goal [7].

Using a top-down hierarchical expansion planning mechanism [12], the text planner plans utterances to achieve discourse goals. When a discourse goal is posted, the text planner searches its library of explanation strategies looking for strategies that can achieve the goal. A strategy is selected and may in turn post subgoals for the planner to refine. Planning continues in this fashion until the entire plan is refined into primitive operators, i.e., speech acts such as **INFORM**, **RECOMMEND**.

As the system plans explanations, it keeps track of any assumptions it makes about what the user knows as well as alternative strategies that could have been used to achieve the discourse goals. The result is a *text plan* for achieving the original discourse goal. This text plan is recorded in the *dialogue history* and passed to the Penman text generation system [4] for translation into English.

²To avoid the myriad problems of parsing English-input (which are being addressed elsewhere, see for example [15]), we assume that the user's questions are posed in a stylized language.

After a response has been generated, the system awaits feedback from the user. This feedback may be a follow-up question (e.g. "Why?", "What's the difference between *setq* and *setf*?"), an indication that the user does not understand the system's response ("Huh?"), an indication that the user understands and has no follow-up question ("Go Ahead"), or a response to a question posed by the system. The query analyzer interprets this feedback and either returns control to the expert system, or formulates the appropriate discourse goal and passes it on to the text planner to produce a response.

If the user asks a follow-up question or indicates that he does not understand the explanation, the system examines the recorded plan in the dialogue history in order to determine how to respond. The information contained there concerning the goal structure of the explanation, assumptions made during its generation, and alternative strategies, is necessary in disambiguating follow-up questions, selecting perspective when describing or comparing objects, and providing further explanations when the user does not ask a well-formulated follow-up question.

ANSWERING A VAGUELY ARTICULATED FOLLOW-UP QUESTION

In Figure 2, we show a sample dialogue with our system. In this example, we illustrate how the system plans the response to the user's question "What is a generalized-variable?" and how the system then elaborates on that response after the user indicates that the response was not understood.

The query analyzer interprets the question and posts the discourse goal (BMB SPEAKER HEARER (KNOW

SYSTEM What characteristics of the program would you like to enhance?

USER Maintainability.

 :

SYSTEM You should replace `(setq x 1)` with `(setf x 1)`.

USER What's the difference between `setf` and `setq`?

SYSTEM `Setf` can be used to assign a value to any generalized-variable. `Setq` can only be used to assign a value to a simple-variable.

USER What is a generalized-variable?

SYSTEM Simple-variables are storage locations named by symbols. This concept can be generalized to a storage location that can be named by any accessor function. These storage locations are called generalized-variables.

USER Huh?

SYSTEM For example, the `car` and `cdr` of a cons are generalized-variables, named by the accessor functions `car` and `cdr`. Other examples are an element of an array or a component of a structure.

Figure 2: Sample Dialogue

HEARER `generalized-variable`)), i.e., the speaker wishes to achieve the state where the speaker and hearer mutually believe that the hearer knows the concept `generalized-variable` [2].

When a discourse goal is posted, the text planner identifies operators capable of satisfying it, i.e., all the operators whose *Effect* matches the goal. The system has several plan operators for achieving the discourse goal (`BMB SPEAKER HEARER (KNOW HEARER ?concept)`) in its library of plan operators. For example, it may describe a concept by describing its attributes and its parts, by drawing an analogy with a similar concept, by giving examples of this concept, or by generalizing a concept the user is familiar with. The plan operator for describing a concept by generalizing is shown in Figure 3.

The planner then attempts to satisfy the *Constraints* of each operator found. The constraints on the operator shown in Figure 3 require that there be a concept, *?sub-concept*, known to the user, that is a subclass of the concept to be described, and another concept, *?super-concept*, that is an immediate superclass of the concept to be described. Those operators whose constraints are satisfied become candidates for achieving the goal.

To choose from among these candidate plan operators, the planner has several selection heuristics, including:

- Prefer operators that require making no assumptions about the hearer's beliefs.
- Prefer operators that make use of a concept the hearer knows.

- Prefer operators that make use of a concept mentioned in the dialogue history.

In this case, the user model indicates that the hearer knows the concept `simple-variable`. Hence the operator in Figure 3 requires making no assumptions about the hearer's knowledge, makes use of a concept the user knows, and uses a concept previously mentioned in the dialogue. Thus it is ranked highest by the plan selection heuristics.

Once a plan operator has been selected, the planner instantiates that plan operator by posting its *Nucleus* and required *Satellites* as subgoals to be refined. An operator for achieving the subgoal (`BMB SPEAKER HEARER (DETAILS-OF ?concept ?attributes ?super-concept)`) appears in Figure 4. `INFORM` is a primitive speech act and the `ELABORATION` subgoal is refined directly into `INFORMs`. This produces the first sentence of the system's response. Space does not permit a detailed description of the plan language, see [7, 6] for more information. In brief, the final text plan for this example first describes `simple-variables` and then abstracts this concept to introduce `generalized-variables`. This produces the response shown in the example dialogue.

The user then indicates that he does not understand this explanation with the vaguely articulated follow-up, "Huh?". From our analysis of naturally occurring dialogues, we devised a set of *recovery heuristics* for responding when the user indicates misunderstanding, but does not ask a well-formulated question. These include:

```

EFFECT: (BMB SPEAKER HEARER (KNOW HEARER ?concept))
CONSTRAINTS: (AND (SUBCLASS ?sub-concept ?concept)
                (BMB SPEAKER HEARER (KNOW HEARER ?sub-concept))
                (IMMEDIATE-SUBCLASS ?concept ?super-concept))
NUCLEUS: ((SETQ ?diffs (ESSENTIAL-DIFFERENCES ?sub-concept ?concept))
           (BMB SPEAKER HEARER (DETAILS-OF ?subconcept ?diffs ?super-concept)))
SATELLITES: (((ABSTRACTION ?sub-concept ?concept ?diffs ?super-concept) *required*))

```

Figure 3: Plan Operator for Describing an Object by Abstraction

```

EFFECT: (BMB SPEAKER HEARER (DETAILS-OF ?concept ?attributes ?super-concept))
CONSTRAINTS: nil
NUCLEUS: (INFORM SPEAKER HEARER (CLASS-ASCRPTION ?concept ?super-concept))
SATELLITES: (((ELABORATION ?concept ?attributes) *required*))

```

Figure 4: Plan Operator for Describing Attributes of an Object

- If the discourse goal is to describe a concept, try giving example(s).
- If the discourse goal is to describe a concept, and there is an analogous entity that the user may be familiar with, draw an analogy to the familiar concept.
- If another plan exists for achieving the discourse goal, try it.
- Expand any unexpanded optional satellites in previous plan operators.³

The first and second heuristics apply in the context of a particular discourse goal, namely describing a concept, while the other heuristics are more general. The system tries to apply its most specific knowledge first. In this case, the first heuristic applies and the explainer recovers by giving examples.

This example brings up another interesting point since it includes the user asking about a term used by the system in a previous response. From our analysis of the data, we found that human experts often use terms that their listeners are not familiar with and that they must later explain. An interesting question is what leads experts to use terms that the listener does not know. Either the expert incorrectly believes that the user knows these concepts, or is making an assumption so that a particular strategy can be used. For example, if the expert has to describe an object that is very complex, and doesn't know whether or not the listener knows the subclass of the object, he may assume that the listener knows the subclass in order to use the (short) strategy of generalizing a concept known to the listener. He then waits for feedback indicating whether or not the explanation was understood.

³Plan operators may contain optional satellites. Depending on other considerations during plan expansion, some of these may not be expanded. See [7].

As illustrated in Figure 3, constraints on plan operators often refer to the state of the hearer's knowledge. The user model includes the domain concepts and problem-solving knowledge, i.e., goals and plans, assumed to be known to the current user. However, we do not assume that this model is either *complete* or *correct*. Therefore, the user model may contain concepts the user does not actually know or omit concepts the user does know.

To satisfy a constraint on an operator, the system may assume that a concept is known to the user even if it is not indicated in the user model. As described above, when such an assumption is made, the selection heuristics give the operator a lower rating. If the operator is selected, the fact that an assumption was made is recorded in the plan structure. The system must keep track of such assumptions because these are likely candidates if a misunderstanding occurs later. This leads to another recovery heuristic:

- If any assumptions were made in planning the last explanation, plan responses to make these assumptions true.

So, for example, in producing the above response about the differences between *setf* and *setq*, the system made the assumption that the user knew what generalized-variables were and simply used this term in the explanation without describing it further. If instead of asking a specific question about generalized-variables in the example dialogue, the user had simply said "Huh" at that point, our system would have planned a response to make this assumption true, i.e., it would have generated a response describing generalized variables.

CURRENT STATUS

The expert system and explanation facility described are implemented. In the current implementation of the

text planner, there are approximately 50 operators, 5 plan selection heuristics, and 5 recovery heuristics. The system can participate in the dialogue shown and several others that are similar in kind. It is also capable of disambiguating and responding to follow-up "why" questions and selecting the correct perspective when comparing objects in the context of an on-going dialogue [7].

CONCLUSIONS

We have argued that an explanation facility must be able to clarify misunderstood explanations and elaborate on previous explanations, even when the user cannot articulate a well-formulated question. However, current systems take an unnatural, one-shot approach to explanation that depends critically on the quality of the user model and is seriously degraded if that model is incomplete or incorrect. These systems do not keep a dialogue history, but even if they did, they do not have alternative strategies for producing responses or heuristics for deciding when different strategies are appropriate.

As an alternative, we proposed a reactive model of explanation – in which the system can employ feedback from the user and participate in a dialogue. Our explanation facility plans responses from a rich set of strategies, keeping track of the system's discourse goals, the plans used to achieve them, and any assumptions made while planning a response. It maintains a history of the text plans it uses to produce responses so that it can later reason about those responses when feedback from the listener indicates that an explanation was not understood. The recovery heuristics presented make use of this history when planning alternative explanations in response to vaguely articulated follow-up questions. Our system can employ information in a user model when it is available, but is not critically dependent on that information.

ACKNOWLEDGEMENTS

The research described in this paper was supported by the Defense Advanced Research Projects Agency (DARPA) under a NASA Ames cooperative agreement number NCC 2-520. I would like to thank William Swartout for guidance on this research, Cécile Paris for collaboration in the design of the plan language, and Eduard Hovy and Yigal Arens for their comments on earlier versions of this paper.

REFERENCES

1. Appelt, D. E. *Planning Natural Language Utterances to Satisfy Multiple Goals*. PhD thesis, Stanford University, 1981.
2. Cohen, P. R., and Levesque, H. J. Speech acts and rationality. In *Proceedings of the Twenty-Third Annual Meeting of the Association for Computational Linguistics* (University of Chicago, Chicago, Illinois, July 8-12 1985), pp. 49-60.
3. Finin, T. W., Joshi, A. K., and Webber, B. L. Natural language interactions with artificial experts. *Proceedings of the IEEE 74*, 7 (July 1986).
4. Mann, W. C., and Matthiessen, C. Nigel: A systemic grammar for text generation. Tech. Rep. RR-83-105, USC/Information Sciences Institute, February 1983.
5. McCoy, K. F. *Correcting Object-Related Misconceptions*. PhD thesis, University of Pennsylvania, December 1985. Published by University of Pennsylvania as Technical Report MS-CIS-85-57.
6. Moore, J. D., and Paris, C. L. Constructing coherent text using rhetorical relations. In *Proceedings of the Tenth Annual Conference of the Cognitive Science Society* (Montreal, Quebec, August 17-19 1988).
7. Moore, J. D., and Swartout, W. R. A reactive approach to explanation, 17-21 July 1988. Presented at the Fourth International Workshop on Natural Language Generation.
8. Neches, R., Swartout, W. R., and Moore, J. D. Enhanced maintenance and explanation of expert systems through explicit models of their development. *IEEE Transactions on Software Engineering SE-11*, 11 (November 1985).
9. Paris, C. L. *The Use of Explicit User Models in Text Generation: Tailoring to a User's Level of Expertise*. PhD thesis, Columbia University, October 1987.
10. Pollack, M. E., Hirschberg, J., and Webber, B. L. User participation in the reasoning processes of expert systems. In *Proceedings of the Second National Conference on Artificial Intelligence* (Pittsburgh, Pennsylvania, August 18-20 1982).
11. Robinson, J. J. Extending grammars to new domains. Tech. Rep. ISI/RR-83-123, USC/Information Sciences Institute, January 1984.
12. Sacerdoti, E. D. A structure for plans and behavior. Tech. Rep. TN-109, SRI, 1975.
13. Sparck Jones, K. User models and expert systems. Tech. Rep. No. 61, University of Cambridge Computer Laboratory, December 1984.
14. Suchman, L. A. *Plans and Situated Actions: The Problem of Human-Machine Communication*. Cambridge University Press, Cambridge, England, 1987.
15. Winograd, T. *Language as a Cognitive Process*, vol. 1: Syntax. Addison-Wesley Publishing Company, 1983.