THE NEED FOR TEXT GENERATION

Kathleen R. McKeown

CUCS-173-85

The need for text generation

by KATHLEEN R. McKEOWN
Columbia University
New York, New York

ABSTRACT

For a variety of systems, such as expert systems, database systems, and problem-solving systems, text generation is one way for the system to communicate effectively with its users. This is particularly true when the system is likely to be used by a wide range of users with varying levels of expertise and background. In this paper I will show why explanation is a crucial feature of expert systems, how text generation can be used within database systems to familiarize users with the database, and where text generation can aid communication with problem-solving systems. Given that text generation is more than just a frill for such systems, a second focus of the paper will be on the kinds of problems that any designer of a text generation system must address. Some of the problems include being able to decide what to say, how to organize that information, and how to express it in natural language.

INTRODUCTION

As computer systems become more sophisticated, they must be able to communicate their results successfully to their users if they are to be effective. For a variety of systems, the use of natural language text is becoming increasingly popular for such communication. This is particularly true when the system is likely to be used by a wide range of users with varying levels of expertise and background. Many potential users of complex computer systems are naive and infrequent users: They not only are unfamiliar with the computer and the formal languages available to interact with it, but their planned use of the system is infrequent enough that it does not warrant the time needed to learn a formal language. For such users, the ability to communicate with systems in everyday language promises to open the door to a world of information and tools they were previously unable to use. For expert users, on the other hand, text generation can provide a summary of complex system behavior that can allow them to detect errors before delving into system details.

In this paper I will show the role text generation plays in three types of systems: expert systems, database systems, and problem-solving systems. These systems all have a wide variety of users and have become complex enough that text generation is appropriate for the interface if ease of use is an issue. Given that text generation is necessary for such systems, the paper will also focus on the problems involved in communicating through natural language. A sketch of the kinds of solutions that have been successfully used is included, but this overview primarily serves as motivation for the text generation problem and an introduction to the solutions and issues considered by the other participants in this session.

EXPERT SYSTEMS

Communication with the user in expert systems has been needed primarily to explain the reasoning used by the system in producing its advice. Textual explanation has proved crucial to the success of expert systems for several reasons.

First, expert system users are often not computer scientists and would be unable to follow a formal representation of the system's reasoning. For example, users of medical expert systems are doctors and medical students. Natural language is a mode of communication familiar to users such as these who may not want to take the time to learn other modes.

Users, though not experts in the programming methodology of expert systems, are often experts in the domain of the system. Again, doctors fit this characterization. Their purpose in using the system is often for consultation: to gain advice on a case or to confirm their own diagnosis. To evaluate the advice provided and to determine whether to accept it, such users need to be able to understand both how and why the system came up with its advice.

Builders and maintainers of expert systems are now pointing out the value of textual explanation in identifying errors in the underlying inferencing process. Often a trace of the inference process itself can be so lengthy (for example, in ACE,[1] a single recommendation may invoke up to 15000 individual production rules) that errors are difficult to detect. Kukich[2] cites this benefit when an explanation facility was first added to the XSEL system.[3] The underlying inferencing system had been constructed incrementally by a number of different researchers who often did not understand the conventions used previously. Her explanation facility immediately pointed out even such simple discrepancies as errors due to roundoff, which had gone undetected.

DATABASE SYSTEMS

Natural-language interfaces to database systems allow a user to retrieve information from the database by asking questions and receiving answers in English. For many questions, the response can be generated by simply formatting the results of a database search in a list or as part of a sentence. It has been shown,[4] however, that many users of database systems, particularly naive and infrequent users, need to ask questions to familiarize themselves with the database before asking specific questions about its contents. Such users need to know what information is available in the database (e.g., "What kind of data do you have?"), what specific terms mean in the context of the database (e.g., "What is production cost?"), or what the differences are between different terms (e.g., "What's the difference between manufacturing and production cost?"). Questions like these typically cannot be answered by doing a search of the underlying database, and this is one place where text generation has played an important role. Natural language is particularly appropriate for answering such questions, since they require definitions, descriptions, and longer textual sequences. To generate these kinds of responses, a formulation of strategies that can be used to organize and determine content of the response is required.[5]

Research in response generation has also addressed the problem of producing responses that cooperatively address the questioner's intentions. Frequently, users reveal in their questions a presumption about the database that turns out to be incorrect. The encoding of presumptions in utterances is a formal feature of natural language that can be exploited to detect and correct a user's misconceptions.[6,7,8] If such presumptions are not corrected, the user may be left with false

beliefs about the database even if all his/her questions have been answered correctly.

PROBLEM-SOLVING SYSTEMS

By problem-solving systems, I mean systems that are capable of working together with another agent (whether a human, another computer, or a robot) to solve a task. Since the two agents must work together cooperatively to solve the problem, communication between them is crucial. In this type of system it is often not necessary to generate lengthy text, but the utterances that are generated must be easily understood by the other participant. For example, in instructing the other participant which tool to use next in a task, the system must use a description that will allow the participant to pick out the correct tool.

Interactive problem-solving systems are often set up so that the system is the expert and the user is an apprentice.[9,10] The user is not knowledgeable about how to perform the task (or solve the problem), and his/her purpose in using the system is to learn how to do so. Some of the tasks that have been considered include construction of a particular piece of equipment (such as the assembly of a water pump) that involves being able to communicate about objects and tools in the physical world. Others have been tasks that could be solved purely verbally, such as in GUS,[11] a system that acts as an airline reservation agent and helps the user select appropriate flights for a trip. Communication in natural language is appropriate for these types of systems, since the user is a novice and thus is unlikely to be familiar with a formal language developed for the domain. Furthermore, language has a long tradition of use as an interactive tool for communication. It is well suited to a situation in which participants must interact heavily to reach a solution.

PROBLEMS IN TEXT GENERATION

The previous sections have demonstrated that there is a need for communication on the part of the system in natural language in a variety of system types. The character of the generated text may differ; but natural language is an appropriate medium, in large part because of the people who will be using such systems. Given that there is a need for text generation in these different types of systems, what types of issues must a designer of a text generation system consider? How do these issues manifest themselves in the different types of systems? How close is text generation research to having practical and implemented solutions to these issues?

In the following sections, a simple but well-tried engineering approach to text generation that is currently used in many systems is first presented. Problems with this approach and issues that must be further considered in order to develop a robust and effective text generation system are then explored.

Canned Text

The simplest method for producing computer-generated text is the use of canned text and templates. This approach is probably the most commonly used method in systems requiring production of a limited amount of text. Most practical expert systems today use templates to produce explanations, and help systems are one of many examples of the use of canned text. The use of canned text requires that the system designer anticipate all questions that might be asked by the user, create the answers by hand, and store these answers as strings, which are retrieved by the system when required.

Templates are slightly more general than canned text, since they provide "text frames" that can be used to answer more than one question of the same type. Templates are English phrases with slots that can be filled in by different words for different occasions. An entire text can be produced by stringing together individual templates that each describe a step in the process. (For example, a single template is associated with each rule in an expert system, and an explanation is produced by stringing together the templates associated with the rules that fired. Slots in the templates are filled by English translations of instantiated variables in the rules.) As with canned text, templates must be produced by the designer by hand when the system is first built, and care must be taken that reasonable texts will be produced when several templates are strung together.

Canned text and templates have the advantage that their generated text can be as sophisticated as the system designer's own prose. Furthermore, it is an engineering technique that is easy to implement. There are numerous problems with this approach, however. Since the system code can be changed independently of the associated text, there is no guarantee that the generated text accurately reflects what the system actually does. An intensive personnel effort is required at the beginning of system development to hand-encode answers, and this effort must be duplicated every time a new system is developed. Finally, in large systems it may be difficult to anticipate all situations in which text will be required in advance.

Deciding What to Say

If text is not prestored ahead of time for the system to retrieve when needed, the text generation module must be able to determine what information to convey, given a request for communication. For certain questions, such as requests for definitions in the database domain, there may be a potentially large amount of information that could be used to answer the question. The system must be able to filter out information in its knowledge base that can be ignored and pinpoint information that should be included.

A number of factors can influence these decisions. The purpose for which information is required can indicate what type of information will be useful. For example, for a request for a definition, information about an object's class membership, its distinguishing attributes, examples, or analogies are appropriate. For a request about the differences between two objects, shared attributes, different class membership, and distinguishing attributes are appropriate. One technique, then, for determining what to say is to use different discourse strategies such as these for different purposes. (For more information, see McKeown.[5])

A text generation system cannot say more than it knows, as represented in its knowledge base. In order to generate particular types of text, it may be necessary to specify the type of information needed in the knowledge base. Swartout[12] shows what information must be added to an expert system knowledge to produce acceptable justifications of the system's advice. Finally, depending on who the system is talking to when a questions is asked, different information will be relevant. Appelt[10] has shown how information about the current user's beliefs should influence what the system says in order to make communication successful. Similarly, Paris[13] identifies how information about the user type (for example, whether naive or expert) can influence how much detail to include in a text.

Deciding When to Say What

Having determined what information is relevant, a text generation system must be able to order that information to produce the text. Order of a text can be crucial to a reader's understanding of it. Order alone can be used to convey temporal sequence, causality, or exemplification. Many early systems simply traced the underlying knowledge base to determine order, doing simple transformations on the underlying data structures. This method requires that the knowledge base be appropriately structured for text generation in addition to meeting all the other demands placed on it. Furthermore, while one knowledge base structure may facilitate inferencing, it may not be appropriate as a blueprint for text production.

Knowledge about discourse structure encoded as strategies can be used for determining order as well as content. Again, for situations where definitions are required, the strategy might not only dictate that class membership information and examples should be used, but also that examples should be included only after class membership has been provided. Discourse structure is currently used to help determine order of presentation in several text generation systems.[5, 13, 14]

Deciding How to Say It

The text generation system must also be able to determine what the surface text should look like. This involves making decisions about what vocabulary to use (and in particular, how to choose between synonyms), when to use a pronoun and when to use a full noun phrase to refer to an object or concept, whether to use a sequence of simple sentences or to combine several simple sentences into a single complex sentence, and how to arrange the words in each sentence. Almost all these decisions are influenced by syntactic constraints on language; thus, one component of a language generation system is a grammar. McDonald[15] describes the kinds of constraints that must be included in a grammar and how it can be used to produce fluent text. One benefit of the use of a grammar over templates is that the system can decide how to combine phrases to produce acceptable text, whereas with templates this work must be done by the system designer manually checking that templates will not produce unacceptable text when strung together.

Other influences on surface level decisions include information about the person the text is intended for and information about the discourse structure of the text. Information about user type can be used to select appropriate vocabulary (the naive user will not understand the expert's terminology). Similarly, information about the user's knowledge can be used to generate noun phrase descriptions so that the user can successfully identify what is referred to by the description.[16] Finally, knowledge about how a given sentence fits in with the rest of the text can be used to choose the best word order for a sentence and to decide whether to use pronouns.

SUMMARY

Decisions that must be made by a text generation system range over a variety of knowledge sources and are influenced by a variety of factors. Furthermore, while I have identified them as separate problems, these problems interact so that decisions often cannot be made independently.[16] Systems currently exist that have addressed each of the problems cited above. Few of these systems, however, handle more than several problems in a single implementation. The other papers in this session focus on specific problems, illustrating more sophisticated text generation techniques that are currently available beyond the limited canned text and template approach.

ACKNOWLEDGMENTS

Research in text generation at Columbia University is partially supported by ARPA grant N00039-84-C-0165 and by ONR grant N00014-82-K-0256.

REFERENCES

1. Stolfo, S., and G. Vesonder. "ACE: An Expert System Supporting Analysis and Management Decision Making." Technical Report, Department of Computer Science, Columbia University, 1982.
2. Kukich, K. "Knowledge-Based Explanation Generation." Paper presented at Second Annual Language Generation Workshop, July 8–10, 1984, Stanford University.
3. McDermott, J. "Building Expert Systems." In *Proceedings of the 1983 NYU Symposium on Artificial Intelligence Applications for Business*. New York: New York University, 1983.
4. Malhotra, A. "Design Criteria for a Knowledge-Based English Language System for Management: An Experimental Analysis." MAC TR-146, Massachusetts Institute of Technology, 1975.
5. McKeown, K. R. "Generating Natural Language Text in Response to Questions about Database Structure." Ph.D. dissertation, University of Pennsylvania, 1982.
6. Kaplan, S. J. "Cooperative Responses from a Portable Natural Language Database Query System." Ph.D. dissertation, University of Pennsylvania, 1979.
7. Mays, E. "Correcting Misconceptions About Data Base Structure." In *Proceedings 3rd Canadian Society for the Computational Studies of Intelligence Biennial Meeting*, Victoria, B.C., 1980.
8. McCoy, K. "Correcting Misconceptions: What To Say When the User is Mistaken." In *Proceedings of Computer and Human Interaction Conference*, 1983 Cambridge, Massachusetts, 1983.
9. Grosz, B. J. "The Representation and Use of Focus in Dialogue Understanding." Technical note 151, Stanford Research Institute, Menlo Park, California, 1977.

10. Appelt, D. E. "Planning Natural Language Utterances to Satisfy Multiple Goals." Ph.D. dissertation, Stanford University, Stanford, California. 1981.
11. Bobrow, D. G., R. M. Kaplan, M. Kay, D. A. Norman, H. Thompson, and T. Winograd. "GUS, A Frame-Driven Dialog System." *Artificial Intelligence*, 8 (1977), pp. 155-173.
12. Swartout, W. "Knowledge Needed for Expert System Explanation." In *AFIPS, Proceedings of the National Computer Conference* (Vol. 54), 1985.
13. Paris, C: "Description Strategies for Naive and Expert Users." Technical Report, Department of Computer Science, Columbia University, 1984.
14. Mann, W. "Discourse Structures for Text Generation." In *Proceedings of Conference on Computational Linguistics*, 1984, Stanford University, 1984.
15. McDonald, D. D. "Surface Generation for a Variety of Applications." *AFIPS, Proceedings of the National Computer Conference* (Vol. 54), 1985.
16. Appelt, D. "Planning and Language Generation in Problem-Solving Systems." In *Proceedings of the National Computer Conference* (Vol. 54), 1985.