NLS-SCHOLAR is a prototype system that uses artificial intelligence techniques to teach computer-naive people how to use a powerful and complex editor. It represents a new kind of computer-assisted instruction (CAI) system that integrates systematic teaching with actual practice, i.e., one which can keep the user under tutorial supervision while allowing him to try out what he learns on the system he is learning about. NLS-SCHOLAR can also be used as an on-line help system outside the tutorial environment, in the course of a user's actual work. This capability of combining on-line assistance with training is an extension of the traditional notion of CAI. The techniques used in NLS-SCHOLAR are general and can be applied to a wide variety of computer related activities. Examples of the system's capabilities are provided. (Author/DGC)
AN "INTELLIGENT" ON-LINE ASSISTANT
AND TUTOR: NLS-SCHOLAR

MARIO C. GRIGNETTI
CATHERINE HAUSMANN
LAURA GOULD

DECEMBER 1974
**AN "INTELLIGENT" ON-LINE ASSISTANT AND TUTOR: NLS-SCHOLAR**

**Abstract**

NLS-SCHOLAR is a prototype system that uses Artificial Intelligence techniques to teach computer-naive people how to use a powerful and complex editor. It represents a new kind of Computer Assisted Instruction (CAI) system that integrates teaching with actual practice, i.e., one which can keep the user under tutorial supervision while allowing him to try out what he learns on the system he is learning about. (See reverse side)
NLS SCHOLAR was developed mainly on a contract with Hanscom Field. Three parts of the project were carried out under this contract: 1) The dialogue analysis on which the design of the system was based, 2) the writing of a primer for tutoring NLS, and 3) the building of LISP-NLS, the version of NLS which the student works with.

NLS-SCHOLAR can also be used as an on-line help system outside the tutorial environment, in the course of a user's actual work. This capability of combining on-line assistance with training is an extension of the traditional notion of CAI.

The techniques used in NLS-SCHOLAR are general and can be applied to a wide variety of computer related activities.
AN "INTELLIGENT" ON-LINE ASSISTANT AND TUTOR: NLS-SCHOLAR

by

Mario C. Grignetti
Catherine Hausmann
Laura Gould

Bolt Beranek and Newman Inc.
Cambridge, Massachusetts

Work described here was sponsored by the AFSC-ESD of Hanscom Field AFB under Contract No. F19628-74-C-0088, and by the Personnel and Training Research Programs, Psychological Sciences Division, Office of Naval Research, under Contract No. N00014-71-C-0228, NR No. 154-330.

This paper has been submitted for presentation at the 1975 National Computer Conference.
ABSTRACT

NLS-SCHOLAR is a prototype system that uses Artificial Intelligence techniques to teach computer-naive people how to use a powerful and complex editor. It represents a new kind of Computer Assisted Instruction (CAI) system that integrates systematic teaching with actual practice, i.e., one which can keep the user under tutorial supervision while allowing him to try out what he learns on the system he is learning about.

NLS-SCHOLAR can also be used as an on-line help system outside the tutorial environment, in the course of a user's actual work. This capability of combining on-line assistance with training is an extension of the traditional notion of CAI.

The techniques used in NLS-SCHOLAR are general and can be applied to a wide variety of computer related activities.
AN "INTELLIGENT" ON-LINE ASSISTANT AND TUTOR: NLS-SCHOLAR

INTRODUCTION

NLS-SCHOLAR is a prototype system that uses Artificial Intelligence techniques to teach computer-naive people how to use the powerful and complex editor of NLS.* This teaching is accomplished by presenting a sequence of lessons. During each lesson the student may interact with the system by asking and answering questions, performing tasks which are posed by the system, and performing tasks of his own choosing. Tasks are actually executed using our own implementation of NLS EDIT.** Those tasks which have been posed are evaluated by the system, and the student is given encouragement, advice, and assistance.

* NLS, the On Line System, is a sophisticated modular system which is being used increasingly as an aid in writing, re-organizing, indexing, publishing, and disseminating information of all kinds [1]. It was developed by Douglas Engelbart and his co-workers at the Augmentation Research Center of the Stanford Research Institute.

** Our system has not yet been interfaced with the real NLS. NLS-SCHOLAR uses LISP-NLS, a partial implementation of NLS's EDIT subsystem written in INTERLISP. The actual interfacing with NLS has been contemplated in LISP-NLS's design, and we hope it will take place in the near future. In the remainder of this paper we shall refer to LISP-NLS as NLS, except where it is important to point out the difference.
NLS-SCHOLAR has been designed with the belief that procedural knowledge is best learned 'by doing' [2,3]. It represents a new kind of Computer Assisted Instruction (CAI) system that integrates systematic teaching with actual practice, i.e., one which can keep a student under "intelligent" tutorial supervision while allowing him to try out what he learns on the very system he is learning about. Thus the system "knows" what the student is doing and can point out his mistakes, give specific help, show him how to do things and even do them for him.

NLS-SCHOLAR is designed so that it can also be used as an on-line help system outside the tutorial environment, allowing users to ask questions arising in their actual work, with NLS-SCHOLAR being aware of what they are doing and answering accordingly. Thus the system can take the lead at first, and fade smoothly into the background as users become proficient. This capability of integrating on-line assistance and training is an extension to the traditional notion of CAI.
Although preserving the flavor and interaction characteristics of SCHOLAR,* NLS-SCHOLAR is an almost entirely new system, its underlying philosophy and approach owing much to Brown's SOPHIE system [7,8]. NLS-SCHOLAR is a prototype for an artificially intelligent system that can offer computer users stand-alone on-line help ranging from occasional assistance to full tutorial guidance and supervision.

*SCHOLAR, conceived and first developed by the late Jaime R. Carbonell, is an interactive mixed-initiative CAI system dealing with the geography of South America [4,5]. It is capable of answering freely interspersed questions posed by the user in the course of a tutorial session, and it uses teaching strategies similar to those of a good human tutor [6].
OVERVIEW

NLS-SCHOLAR has the following capabilities:

a) When used in tutorial mode, it delivers a series of lessons designed for gradual understanding of NLS concepts and commands. Within these lessons, the system pauses to ask the student questions and to propose editing tasks for him to perform using NLS. A student’s responses to questions and his performance of tasks are evaluated by the system and if he makes an error, the nature of his mistake is pointed out and appropriate action taken. For example, if a question is answered unsatisfactorily, NLS-SCHOLAR proposes another question of the same kind. If a task is performed incorrectly, depending on the magnitude of the error, NLS-SCHOLAR either resets it for the student to try again; or asks him to proceed and try to fix his mistake, aided by the information NLS-SCHOLAR provides.

b) The user can formulate requests in relatively unconstrained English. The requests can be questions about NLS concepts or about the state of his work, requests for help in doing a task, or even NLS commands expressed in English. The system is "aware" of what the user is currently doing so that his requests for help can be answered within the context of the problem he is working on. Thus NLS-SCHOLAR not only tells him "The general procedure is..." but also "In your case, what you should do is...".

c) NLS-SCHOLAR has the ability to use a person’s work space (the NLS file he is currently working on) to show him how to perform editing actions. This gives the system much of the flavor of a human tutor, as if he were taking the student’s place at the terminal and saying "Watch me do it for you".

d) NLS-SCHOLAR is very friendly. Students can ask questions whenever it is their turn to type, make mistakes safely, ask for help doing tasks, and give up and be rescued by the system.

These capabilities allow people to learn from explanation, learn by doing, and learn by asking questions. Their tight integration within a working environment makes NLS-SCHOLAR a powerful assistant to its users.
DEMONSTRATING NLS-SCHOLAR'S CAPABILITIES

The flavor of NLS-SCHOLAR is best conveyed by an annotated demonstration protocol which was actually obtained on-line using the latest version of the system. First a few helpful comments:

It is difficult to give a demonstration of a system's capabilities "in vacuo"; questions asked by a student or by the system, as well as tasks proposed and evaluated, arise more naturally and make more sense in the course of a lesson. Since this is a demonstration protocol, our "student" (actually one of the authors) is very obliging and does the appropriate things at the right times to demonstrate specific characteristics of the system.

NLS-SCHOLAR uses two bodies of text as its working examples, one a breakfast menu (see figure 1) and the other a dinner menu. In the course of a lesson, students learn how to change the contents (and appearance) of these menus by performing editing operations. Menus were chosen as examples because of their direct appeal and general intelligibility, and because the shortness of their entries makes them easy to work with.
In the interest of brevity,* the protocol starts at a point well along in the student's learning of NLS -- he has been told about NLS files, how to load them, print them, delete and insert statements, etc. He is about to be taught how to use the Substitute command to effect a change in the breakfast menu. Notice that what the student actually typed is underlined and that our annotations appear in italics.

Readers familiar with NLS may fail to recognize it as the system depicted in the protocol. This is because NLS-SCHOLAR teaches the use of a newly emerging version of NLS not yet generally available.

*For a complete demonstration of the present capabilities of NLS-SCHOLAR, see the protocols included in reference [2].
<NLS-SCHOLAR>BREAKFAST.LNLS;1 1-OCT-74 03-50 CLH ;
1 JUICE
1A ORANGE
1B GRAPEFRUIT
2 CEREAL
2A OATMEAL
2A1 WITH RAISINS
2B CREAM OF WHEAT
2C CORN FLAKES
3 EGGS
3A SCRAMBLED
3B FRIED
3B1 SUNNY-SIDE-UP
3B2 OVER-EASY
3C BOILED
4 BEVERAGE
4A HOT CHOCOLATE
4B TEA
4B1 WITH LEMON
4B2 WITH SUGAR AND CREAM
4C COFFEE

Figure 1: The BREAKFAST File
SUBSTITUTE WORD IN STATEMENT

Before we end this lesson, I'd like to show you how to change the content of statements which have already been inserted in your file.

You've learned how to correct errors by deleting an entire statement and inserting a new one in its place. The Substitute command may be used to change just a word or even a single character at a time. For example, here's how you can change a word in statement 2A1 so that it reads WITH DATES instead of WITH RAISINS.

EDIT C: (Su) bstitute C: (W) ord in C: (S) tatement at A: (2A1) (<CR>);
   (New WORD) T: (DATES) (<CR>);
   (Old WORD) T: (RAISINS) (<CR>);
   Finished? Y/N: (Y) OK: (<CR>);
   Substitutions made: 1
EDIT C: (Do) ne OK: (<CR>);

NLS-SCHOLAR performs a command for the student much as a human tutor would by loaning over the student and typing the command on the student's terminal. Parentheses enclose what the "tutor" actually "types", separating it from the echoes and responses printed by NLS. <CR> is just a symbol to denote that the "tutor" depressed the return key. The semicolon is NLS's echo to <CR>.

You can see that I've specified that I want to substitute one word in statement 2A1 for another; the new word DATES is to be substituted for the old word RAISINS.
(There is no restriction that the items substituted for one another be of the same length.)

To do this I typed only 'Su W S 2A1 <CR>' to produce the first line; I typed 'DATES <CR>' and 'RAISINS <CR>' to produce the second and third. The remaining characters are all printed by NLS.

NLS asks 'Finished?' since more than one substitution can be made at a time. If I had typed 'N' for no instead of 'Y' for yes, it would have prompted for another <New WORD> and <Old WORD> pair.

After I typed 'Y' NLS asks for further confirmation with OK: to which I responded with a <CR>.

When the command has been executed, NLS tells how many substitutions have been made. If there had been two instances of RAISINS in statement 2A1, both would have been changed to read DATES.

Statement 2A1 is now the current statement. Please print it.
The student confuses the \ command, which prints the current statement, with the / command that shows the current position of the control marker. Realizing his mistake, he fumbles with the other one-character commands he can remember. First he tries t, which prints the previous statement (2A), and makes it become the current statement.

Next the student depresses the LINE FEED key which echoes as %, giving him the next statement (2A1), making it current.

You printed too many statements. This is what I wanted you to print:

2A1 WITH DATES

Please redo just the printing part of this task. Your file is unchanged.

NLS-SCHOLAR is really determined to get the student to review how to print the current statement.

continuation of TASK 41

EDIT C: *What command prints the current statement?*

THE USER PRINTS THE CURRENT STATEMENT USING THE \ COMMAND.

Failing to remember the appropriate one-character command, the student gives up and asks for help. Questions can be asked anytime it is the student's turn to type. NLS-SCHOLAR answers this question by retrieving information from its own internal, descriptive data base.

continuation of TASK 41

EDIT C: \ 2A1 WITH DATES
EDIT C: Done_OK:_;

QUESTIONS?

** The student chooses to ask no questions at this time.
Try the Substitute command yourself by substituting the word OAT for CORN in the statement containing "CORN FLAKES".

TASK 42
EDIT C: *What statement contains "CORN"?*

THE STATEMENT CONTAINING "CORN" IS:
2C CORN FLAKES

The student wants to know what the correct statement number is for him to type. The system obtains the answer by actually looking into the student's work space -- the answer is not stored (it cannot be!) in the internal data base.

The system interprets correctly most of the common paraphrases of this type of question, e.g.
"What is the statement containing ..."
"What statement contains ..."
"Where is ...", etc.

continuation of TASK 42
EDIT C: Substitute C: Word in C: Statement at A: 2C;
<New WORD> T: CORN;
<Old WORD> T: OAT;
Finished? Y/N: Y OK:_;
Substitutions made: 0
EDIT C: Done OK:_;

The student switches around the new and the old words, so no change is made.

You should have altered the file so that this part of it
2B CREAM OF WHEAT
2C CORN FLAKES
3 ...

would look like this:
2B CREAM OF WHEAT
2C OAT FLAKES
3 ...

But you did not alter the file.

Please try to repair your mistake. The file is as you left it.
continuation of TASK 42

EDIT C: Substitute C: Word in C: Statement at A: 2C:
<New WORD> T: OAT;
<Old WORD> T: CORN;
Finished? Y/N: Y OK: ;
Substitutions made: 1
EDIT C: Done OK: ;

Now print your result.

TASK 43

EDIT C: Print OK:/C: ;
  2C OAT FLAKES
  3 EGGS
    3A SCRABBLED
    3B FRIED
      3B1 SUNNY-SIDE-UP
      3B2 OVER-EASY
    3C BOILED
  4 BEVERAGE
    4A HOT CHOCOLATE
    4B TEA
      4B1 WITH LEMON
      4B2 WITH SUGAR AND CREAM
    4C COFFEE
EDIT C: Done OK: ;

You printed too many statements.
This is what I wanted you to print: 2C OAT FLAKES

Please try to print what I wanted. The file is as you left it.

continuation of TASK 43

EDIT C: \  
  2C OAT FLAKES
EDIT C: Done OK: ;

Good. The noun 'WORD' refers to the fact that whole words are involved, not just parts of words; but it does not imply that a single word only may be provided as either the <New WORD> or the the <Old WORD>. Use the Substitute command to change statement 2A1 again, this time to read WITH BROWN SUGAR AND CREAM instead of WITH DATES and print your result.
TASK 44

EDIT C: Substitute C: Word in C: Statement at A: 2A1;
    <New WORD> T: WITH BROWN SUGAR AND CREAM;
    <Old WORD> T: DATES;
Finished? Y/N: Y OK:__;
Substitutions made: 1
EDIT C: Done OK:__;

I wanted you to change
this part of your file:
    2A OATMEAL
    2A1 WITH DATES
    2B . . .
into this:
    2A OATMEAL
    2A1 WITH BROWN SUGAR AND CREAM
    2B . . .

But you changed this part of it:
    2A OATMEAL
    2A1 WITH DATES
    2B . . .
into this:
    2A OATEMAL
    2A1 WITH WITH BROWN SUGAR AND CREAM
    2B . . .

I have restored the file as it was
at the beginning of this task. Please start over.
TASK 44

EDIT C: *How does one do this?* The student is lost and asks for help.


*NLS-SCHOLAR can show him what to do, as well as...*

continuation of TASK 44

EDIT C: *Please do it for me* ...actually do it for him!

EDIT C: (Substitute ) C: (Word ) in C: (Statement ) at A: (2A1 )(<CR>);
  <New WORD> T: (BROWN SUGAR AND CREAM)(<CR>);
  <Old WORD> T: (DATES)(<CR>);
  Finished? Y/N: (Y) OK: (<CR>);
  Substitutions made: 1
EDIT C: (\ )
  2A1 WITH BROWN SUGAR AND CREAM
EDIT C: (Done ) OK: (<CR>);

QUESTIONS?

*NLS* The student can summon NLS in order to practice (or just to play) without fear of messing up his work.

...USING A DISPOSABLE COPY OF YOUR CURRENT FILE

EDIT C: Print OK:/C: Statement at A: 2B;
  V: .
  2B CREAM OF WHEAT
EDIT C: Substitute C: Character in C: Statement at A: 2B;
  <New CHARACTER> T: U;
  <Old CHARACTER> T: EA;
  Substitutions made: 2
EDIT C: \ 2B CRUM OF WHUT

The student replaced the characters EA for U.
EDIT C: Print OK:/C: Branch at A: "EGGS";
V:_;
3 EGGS
  3A SCRAMBLED
  3B FRIED
    3B1 SUNNY-SIDE-UP
    3B2 OVER-EASY
  3C BOILED
EDIT C: Substitute C: Character in C: Branch at A: 3;
<New CHARACTER> T: *;
<Old CHARACTER> T: _;
Finished? Y/N: Y OK:_;
Substitutions made: 3
EDIT C: Print OK:/C: C: Branch at A: 3B;
V:_;
3B FRIED
    3B1 SUNNY-SIDE*UP
    3B2 OVER*EASY
EDIT C: Done OK:_;

...THROWING AWAY THIS COPY

The student leaves NLS returning to the 'QUESTIONS?' level. The "Transparent overlay" on which he has scribbled disappears without trace.

*Print branch_3B, please* Commands can be issued in natural language.

...USING A DISPOSABLE COPY OF YOUR CURRENT FILE

EDIT C: (Print ) OK:/C: (Branch ) A: (3B) (<CR>);
V: (<CR>);
  3B FRIED
    3B1 SUNNY-SIDE-UP
    3B2 OVER-EASY
EDIT C: (Done ) OK: (<CR>);

...THROWING AWAY THIS COPY
REPRESENTATION OF KNOWLEDGE

Much of NLS-SCHOLAR's knowledge is derived from data stored in a semantic network, and from a set of built-in routines that manipulate and retrieve that data in response to queries. The semantic network is a data base of descriptive information represented in attribute-value format. It contains descriptions of actions and their purposes, descriptions of the procedures necessary to accomplish those actions, and descriptions of their effects and consequences. For example, the semantic network contains the description of the purpose of the Delete command as well as the description of the procedure for its use. English renditions of these attribute-value representations are: "The purpose of the Delete command is to delete a structure unit", and "The procedure (for deleting a structure unit) is for the user to type the word DELETE, followed by the name of the structure unit, the address, and two carriage returns".

The semantic network also contains many other kinds of representations, among them the definitions of concepts, the interrelationships between concepts (such as that a statement is an instance of a structure unit), and the sequence of commands necessary to perform each task correctly.
The retrieval routines, taking a user's query as their starting point, look into the semantic network seeking information relevant to that query. For example, if a user wants to know what the line-feed command is used for, his question would translate into a query that would essentially mean: "Find the purpose of the line-feed command". The retrieval routines would attempt several different matching procedures that would eventually yield something like: "The purpose of the line-feed command is to print the next statement".

The retrieval process is assisted by built-in "reasoning" strategies which are called upon when the matching procedures fail. In fact, in many cases the desired information is not directly stored, but can be inferred from available information. For example, if the query were for the procedure for deleting a statement, the matching procedures would fail. However, the retrieval system would still be able to derive the answer via simple deductive inference: it knows that a statement is a kind of structure unit, and it knows how to delete structure units, therefore it can derive the procedure "Type 'DELETE', followed by 'STATEMENT', ...".

Mechanisms such as the ones just described are the seat of the abstract "thinking" abilities of NLS-SCHOLAR. As such, they are not yet very powerful, and much can be done
to improve them.* However, it is important to stress here that there is more to "intelligence" than powerful manipulation of symbols.

People's intelligent behavior is not based solely on internal representations and conceptualizations and their attendant reasoning procedures. People's data bases are not only in their memories, neither are their retrieval "routines" solely introspective. We use the world as a data base, and our senses to retrieve information from it. I don't need to have in my head a representation of what is behind my chair; if I need to know, I can just turn around, look, and see!

Because NLS-SCHOLAR deals with a "world" (the world of NLS) with which it shares much of its own being (i.e., it is a computer program that deals with another computer program), it was relatively easy to endow it with some of this latter kind of "intelligence". For example, to make NLS-SCHOLAR "aware" of the state of a user's work, all we had to do was design it so that it could couple with NLS and use it as a sort of sensor of the "world" of the user's file.

*Much work has been done on this problem in the SCHOLAR system dealing with the geography of South America [9].
This coupling of two systems (NLS-SCHOLAR and NLS itself) constitutes an exceedingly powerful tool. First, it makes it possible for the user to ask questions not only about definitions, descriptions of procedures, etc. (such as "What does back statement mean?", "What command prints the back statement?", or "How do I print a file?"), but also about the ever-changing state of his work (such as "What is the content of statement 3A?", or "Where is the CH now?" or "Print just branch 3 for me"). Thus, in addition to searching for answers in a static semantic network we gain the ability to interrogate the dynamic "NLS world" as well.

Second, this coupling provides an easy way of performing a type of "if-then" inference that would be very hard to perform deductively. Suppose a user asked something like

"If I deleted statement 2B, what would then be the statement number of the statement containing "COEN"?

Finding the answer by deductive reasoning is possible but difficult. Obtaining the answer by using NLS and "sotto voce" deleting statement 2B and then seeing where the statement containing "COEN" ends up illustrates a powerful use of this coupling.*

*In fact, a new breed of "intelligent" CAI systems based on this approach has been pioneered by Brown and his co-workers [8].
Third, it becomes possible to propose problems or tasks to a student and to evaluate his solutions in an interesting way. All the system has to do is access the correct sequence of NLS commands for the task, perform them on a fresh copy of the student’s file, and then compare the results.

Lastly, NLS-SCHOLAR can use its semantic network and reasoning routines to infer a procedure (such as how to delete a statement), use this information to construct an NLS command, and then execute that command. Thus it is able not only to describe procedures but also to synthesize NLS commands using this knowledge.

OVERALL ORGANIZATION

The overall organization of NLS-SCHOLAR is represented in Figure 2. There is an EXECUTIVE which controls and supervises the main functions of the system (question answering, question asking, text delivery, and task monitoring), services their requests, and provides communication paths among them. When in tutorial mode, EXECUTIVE is driven by an AGENDA containing general instructions of what to do next (deliver text, perform a task as if a tutor were demonstrating how to do it, answer questions, evaluate a student’s answers, etc.).
Figure 2. SYSTEM ORGANIZATION
TASK MONITOR decides how to call NLS. It can simply allow users to type their commands directly into LISP-NLS, it can make use of "tutor typing" of commands either retrieved from the database or synthesized by Q/ANSWER, or it can have these commands executed invisibly to the user.

Q/ANSWER is a facility for responding to a user's requests. Q/ANSWER responds not only to questions whose answers are static (i.e. retrievable from the semantic network as in "Give me some examples of printing commands"), but also to questions which refer to what a user is doing and which have answers that are dynamic, i.e., that change with time. For example, the question

WHERE IS THE "CORN"?

must be interpreted as a call to NLS to find the address of the word "CORN" as it exists in the current file. To do this, Q/ANSWER has to synthesize the appropriate NLS commands

\[
\begin{align*}
\text{Jump Statement 0 } & \text{<CR>} \\
\text{Jump "CORN" } & \text{<CR>}
\end{align*}
\]

have the context manipulation machinery save the user's environment, perform the commands invisibly, restore the user's environment, and hand back the result of executing these commands to Q/ANSWER which then generates a response.

The parser in Q/ANSWER is an adaption of the top-down, semantically directed parser described by Burton [10],
augmented with capabilities that analyze and label the Case relationships [11,3] existing between the main verb and the noun phrases of an input request. In addition it determines the general category that the request falls into (a request for a definition, procedure, address of some word in the current file, etc.). For example, the question

HOW DO I PRINT BRANCH 3A?

parses into the form

(QFIND/PROCEDURE ((AGENT USER) (VERB PRINT) (OBJ BRANCH (ADDR 3A))))

Thus the interpretation of a request is a LISP function which can then be evaluated (executed) to retrieve an answer. That is, QFIND/PROCEDURE is a LISP function that takes a Case parsed sentence as its input, retrieves the correct response from the data base, and outputs the answer.

In our case, the answer is:

THE USER PRINTS BRANCH 3A USING THE PRINT COMMAND.

Only the general procedure for printing a structure unit is in the semantic network. Q/ANSWER instantiates elements specific to the request, in this case the specific structure unit 'BRANCH' and the specific address '3A' (but not the specific viewspecs because they were not stated in the request).

Q/ASKING is a facility that generates questions for the
user to answer. (Although this is not demonstrated in the protocol, the system occasionally asks questions as well as answering them.) If the student's answer is found to be correct, the lesson continues. If it is only partially correct, the part in error is pointed out and another question of the same type is generated. The student does not have to answer the question immediately; he can ask questions and obtain answers from the system, which will then remind him of the question it asked that is still unanswered. The following sample dialogue illustrates this facility. (Parts typed by the user are underlined.)

WHAT STATEMENTS ARE IN BRANCH 3B?

*3B1 and 3B2*

CORRECT. HOWEVER, YOUR ANSWER IS INCOMPLETE. 3B1 AND 3B2 ARE O.K. YOU COULD ALSO INCLUDE 3B.

WHAT STATEMENTS ARE IN BRANCH 1?

*Give me a definition of a branch*

A BRANCH IS A STATEMENT PLUS ALL ITS SUBSTATEMENTS.

NOW PLEASE ANSWER MY QUESTION.

*1, 1A, and 1B*

VERY GOOD

Observe that only the question type is preselected; Q/ASKING knows how to select valid branches right out of the user's file, and to generate questions accordingly.
The commands used by the student.
2) A representation of the structure and contents of the file.
3) The position of the control marker.
4) The state of the "viewspecs" (what kind of a "viewing filter" was used to view the file).
5) What the student actually printed.

Each of these components defines a "specialist-reporter": a program that knows how to diagnose errors within its specialty and how to indicate to the student what was wrong. For example, the File Structure specialist-reporter compares the result of a student's execution of a task with the correct result and classifies the discrepancies into three cases: something missing, something extra, or something that has a different content. In each case, only enough information to show the discrepant parts is shown to the student ("I wanted you to change xx into yy, but instead you changed zz into tt").

CONCLUSIONS

As computer systems grow in power, sophistication, and complexity, it becomes more and more difficult to become (or even remain) an expert in their usage. Many users prefer sticking to the outdated but familiar facilities offered by a new upward compatible system rather than learning to use
the new, more powerful facilities. With the advent of large, geographically dispersed computer facilities, it becomes more and more difficult to get hold of the resident expert and ask him to look over one's problems. There is a real need for something to take these experts' places. We believe that the class of "intelligent" on-line assistants and tutors of which NLS-SCHOLAR is a prototype are a promising solution to this problem.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the work of Alan G. Bell who wrote LISP-NLS, of Gregory Harris who was responsible for the specialists-reporters, and of Joseph Passafiume who wrote the question asking module. This work would not have been possible without the contagious enthusiasm that emanated from countless hours of discussion with John Seely Brown. To him, a very special "thank you".
REFERENCES


DISTRIBUTION LIST

4 Dr. Marshall J. Farr, Director
Personnel & Training
Research Programs
Office of Naval Research
Arlington, VA 22217

1 Director
ONR Branch Office
495 Summer Street
Boston, MA 02210
ATTN: C. M. Harsh

1 Director
ONR Branch Office
1030 East Green Street
Pasadena, CA 91101
ATTN: E. E. Gloye

1 Office of Naval Research
Area Office
207 West 24th Street
New York, NY 10011

2 Director
Advanced Research Projects
Agency
1400 Wilson Boulevard
Arlington, VA 22209
Attention: Program Management
DODAAAD Code HX1241

1 STOIAC
Battelle Memorial Institute
505 King Avenue
Columbus, Ohio 43201
Attention: H8-1 Code 79986

6 Director
Naval Research Laboratory
Code 2627
Washington, DC 20390

12 Defense Documentation
Center
Cameron Station,
Building 5
5010 Duke Street
Alexandria, VA 22314

1 Chairman
Behavioral Science Department
Naval Command and Management
Division
U.S. Naval Academy
Luce Hall
Annapolis, MD 21402

1 Chief of Naval Technical
Training
Naval Air Station Memphis(75)
Millington, TN 38054
ATTN: Dr. G. D. Mayo

1 Chief of Naval Training
Naval Air Station
Pensacola, FL 32508
ATTN: CAPT Bruce Stone, USN

1 TACTEC
Battelle Memorial Institute
505 King Avenue
Columbus, Ohio 43201
Attention: H8-1 Code 79986

1 Defense Contract Administration
Services Region
666 Summer Street
Boston, MA 02210
Attention: DODAAAD Code 52202A
1 Mr. Arnold Rubinstein  
Naval Material Command  
(NMAT-03424)  
Room 820, Crystal Plaza #6  
Washington, D.C. 20360

1 Superintendent  
Naval Postgraduate School  
Monterey, CA 92940  
ATTN: Library (Code 2124)

1 Technical Library  
Naval Ship Systems Command  
National Center, Building 3  
Room 3S08  
Washington, D.C. 20360

1 Chief of Naval Training Support  
Code N-21, Building 45  
Naval Air Station  
Pensacola, FL 32508

1 Dr. H. Wallace Sinaiko  
c/o Office of Naval Research  
(Code 450)  
Psychological Sciences Division  
Arlington, VA 22217

1 Commandant  
U.S. Army Institute of Administration  
ATTN: EA  
Fort Benjamin Harrison, IN 46216

1 Armed Forces Staff College  
Norfolk, VA 23511  
ATTN: Library

1 Commander  
U.S. Theatre Army Support Command, Europe  
ATTN: Asst. DCSPER (Education)  
APO New York 09058

1 Commanding Officer  
Naval Personnel Research and Development Center  
San Diego, CA 92152

1 Mr. George N. Graine  
Naval Ship Systems Command  
(SHIPS 03H)  
Department of the Navy  
Washington, D.C. 20360

1 Commanding Officer  
Service School Command  
U.S. Naval Training Center  
San Diego, CA 92133  
ATTN: Code 303

1 Dr. William L. Maloy  
Principal Civilian Advisor for Education and Training  
Naval Training Command, Code 01A  
Pensacola, FL 32508

1 Dr. Martin F. Wiskoff  
Navy Personnel Research and Development Center  
San Diego, CA 92152

1 Chief, Unit Training and Educational Technology Systems  
U.S. Army Research Institute for the Behavioral and Social Sciences  
1300 Wilson Boulevard  
Arlington, VA 22209

1 Director of Research  
U.S. Army Armor Human Research Unit  
ATTN: Library  
Building 2422 Morade Street  
Fort Knox, KY 40121
<table>
<thead>
<tr>
<th>1</th>
<th>Commandant, Marine Corps</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Code AO1M-2</td>
</tr>
<tr>
<td></td>
<td>Washington, D.C. 20380</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>COL George Caridakis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Director, Office of</td>
</tr>
<tr>
<td></td>
<td>Manpower Utilization</td>
</tr>
<tr>
<td></td>
<td>Headquarters, Marine</td>
</tr>
<tr>
<td></td>
<td>Corps (AO1H), MCB</td>
</tr>
<tr>
<td></td>
<td>Quantico, VA 22134</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>Dr. A. K. Slafkosky</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Scientific Advisor (Code Ax)</td>
</tr>
<tr>
<td></td>
<td>Commandant of the Marine Corps</td>
</tr>
<tr>
<td></td>
<td>Washington, DC 20380</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>Mr. E. A. Dover</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Manpower Measurement Unit (Code AOIM-2)</td>
</tr>
<tr>
<td></td>
<td>Arlington Annex, Room 2413</td>
</tr>
<tr>
<td></td>
<td>Arlington, VA 20370</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>Mr. Joseph J. Cowan, Chief</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Psychological Research Branch (P-1)</td>
</tr>
<tr>
<td></td>
<td>U. S. Coast Guard Headquarters</td>
</tr>
<tr>
<td></td>
<td>400 Seventh Street, SW</td>
</tr>
<tr>
<td></td>
<td>Washington, DC 20590</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>Dr. John Ford, Jr.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Navy Personnel Research and Development Center</td>
</tr>
<tr>
<td></td>
<td>San Diego, CA 92152</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>LCDR Charles Theisen, Jr.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSC, USN, 4024</td>
</tr>
<tr>
<td></td>
<td>Naval Air Development Center</td>
</tr>
<tr>
<td></td>
<td>Warminster, PA 18974</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>Mr. Helga Yeich, Director</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Program Management, Defense Advanced Research Projects Agency</td>
</tr>
<tr>
<td></td>
<td>1400 Wilson Boulevard</td>
</tr>
<tr>
<td></td>
<td>Arlington, VA 22209</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>Mr. William J. Stormer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DOD Computer Institute</td>
</tr>
<tr>
<td></td>
<td>Washington Navy Yard</td>
</tr>
<tr>
<td></td>
<td>Building 175</td>
</tr>
<tr>
<td></td>
<td>Washington, DC 20374</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>Dr. Ralph R. Canter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Director for Manpower Research</td>
</tr>
<tr>
<td></td>
<td>Office of Secretary of Defense</td>
</tr>
<tr>
<td></td>
<td>The Pentagon, Room 3C980</td>
</tr>
<tr>
<td></td>
<td>Washington, DC 20301</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>Office of Computer Information</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Institute for Computer Sciences and Technology</td>
</tr>
<tr>
<td></td>
<td>National Bureau of Standards</td>
</tr>
<tr>
<td></td>
<td>Washington, DC 20234</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>Dr. Eric McWilliams</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Program Manager</td>
</tr>
<tr>
<td></td>
<td>Technology and Systems, TIE</td>
</tr>
<tr>
<td></td>
<td>National Science Foundation</td>
</tr>
<tr>
<td></td>
<td>Washington, DC 20550</td>
</tr>
</tbody>
</table>
1 Dr. Scarvia Anderson
Executive Director for Special
Development
Educational Testing Service
Princeton, NJ 08540

1 Dr. Richard C. Atkinson
Stanford University
Department of Psychology
Stanford, CA 94305

1 Dr. Bernard M. Bass
University of Rochester
Management Research Center
Rochester, NY 14627

1 Mr. Edmund C. Berkeley
Berkeley Enterprises, Inc.
815 Washington Street
Newtonville, MA 02160

1 Dr. David G. Bowers
University of Michigan
Institute for Social Research
P. O. Box 1248
Ann Arbor, Michigan 48106

1 Dr. Ronald P. Carver
American Institutes
for Research
8555 Sixteenth Street
Silver Spring, MD 20910

1 Century Research Corporation
4113 Lee Highway
Arlington, VA 22207

1 Dr. Robert Glaser, Director
University of Pittsburgh
Learning Research and
Development Center
Pittsburgh, PA 15213

1 Dr. Kenneth E. Clark
University of Rochester
College of Arts and Sciences
River Campus Station
Rochester, NY 14627

1 Dr. Rene' V. Dawis
University of Minnesota
Department of Psychology
Minneapolis, MN 55455

2 ERIC
Processing and Reference
Facility
4833 Rugby Avenue
Bethesda, MD 20014

1 Dr. Victor Fields
Department of Psychology
Montgomery College
Rockville, MD 20850

1 Mr. H. Dean Brown
Stanford Research Institute
333 Ravenswood Avenue
Menlo Park, CA 94025

1 Mr. Michael W. Brown
Operations Research, Inc.
1400 Spring Street
Silver Spring, MD 20910

1 Dr. Edwin A. Fleishman
American Institutes for
Research
8555 Sixteenth Street
Silver Spring, MD 20910

1 Dr. Albert S. Glickman
American Institutes
for Research
8555 Sixteenth Street
Silver Spring, MD 20910
Dr. Duncan N. Hansen  
Florida State University  
Center for Computer-Assisted Instruction  
Tallahassee, FL 32306

Dr. Henry J. Hamburger  
University of California  
School of Social Sciences  
Irvine, CA 92664

Dr. Richard S. Hatch  
Decision Systems Associates Inc.  
11428 Rockville Pike  
Rockville, MD 20852

Dr. M. D. Havron  
Human Sciences Research, Inc.  
Westgate Industrial Park  
7710 Old Springhouse Road  
McLean, VA 22101

Human Resources Research Organization  
Division #3  
P. O. Box 5787  
Presidio of Monterey, CA 93940

Human Resources Research Organization  
Division #4, Infantry  
P. O. Box 2086  
Fort Benning, GA 31905

Human Resources Research Organization  
Division #5, Air Defense  
P. O. Box 6057  
Fort Bliss, TX 79916

Human Resources Research Organization  
Division #6, Library  
P. O. Box 428  
Fort Rucker, AL 36360

Dr. Lawrence B. Johnson  
Lawrence Johnson and Associates, Inc.  
200 S Street, NW, Suite 502  
Washington, DC 20009

Dr. Norman J. Johnson  
Carnegie-Mellon University  
Graduate School of Industrial Admin.  
Pittsburgh, PA 15213

Dr. David Klahr  
Carnegie-Mellon University  
Graduate School of Industrial Admin.  
Pittsburgh, PA 15213

Dr. Robert R. Mackie  
Human Factors Research, Inc.  
6780 Cortona Drive  
Santa Barbara Research Park  
Goleta, CA 93017

Dr. Andrew R. Molnar  
Technological Innovations in Education  
National Science Foundation  
Washington, DC 20550

Dr. Leo Munday  
Vice President  
American College Testing Program  
P.O. Box 168  
Iowa City, IA 52240

Dr. Donald A. Norman  
University of California, San Diego  
Center for Human Information Processing  
La Jolla, CA 92037

Mr. Luigi Petrullo  
2431 North Edgewood Street  
Arlington, VA 22207
Dr. Robert D. Pritchard  
Assistant Professor of Psychology  
Purdue University  
Lafayette, IN  47907

Dr. Diane M. Ramsey-Klee  
R-K Research & System Design  
3947 Ridgemont Drive  
Malibu, CA  90265

Dr. Joseph W. Rigney  
Behavioral Technology Laboratories  
University of Southern California  
3717 South Grand  
Los Angeles, CA  90007

Dr. Leonard L. Rosenbaum, Chairman  
Department of Psychology  
Montgomery College  
Rockville, MD  20850

Dr. George E. Rowland  
Rowland and Company, Inc.  
P. O. Box 61  
Haddonfield, NJ  08033

Dr. Arthur I. Siegel  
Applied Psychological Services and Science Center  
404 East Lancaster Avenue  
Wayne, PA  19087

Mr. Dennis J. Sullivan  
725 Benson Way  
Thousand Oaks, CA  91360

Mr. Dexter Fletcher  
Department of Psychology  
University of Illinois  
at Chicago Circle  
Box 4348  
Chicago, IL  60680

Dr. Benton J. Underwood  
Northwestern University  
Department of Psychology  
Evanston, IL  60201

Dr. David J. Weiss  
University of Minnesota  
Department of Psychology  
Minneapolis, MN  55455

Denver Research Institute  
University of Denver  
Denver, CO  80210

Dr. Kenneth Wexler  
University of California  
School of Social Sciences  
Irvine, CA  92664

Dr. John Annett  
The Open University  
Milton Keynes  
Buckinghamshire  
ENGLAND

Dr. Milton S. Katz  
MITRE Corporation  
Westgate Research Center  
McLean, VA  22101

Dr. Charles A. Ullmann  
Director, Behavioral Sciences Studies  
Information Concepts Incorporated  
1701 No. Ft. Myer Drive  
Arlington, VA  22209

Dr. Anita West  
Denver Research Institute  
University of Denver  
Denver, CO  80210