Providing Help and Advice in Task Oriented Systems

Timothy \V. Finin

Department of Computer and Information Science
The Moore School
University of Pennsylvania
Philadelphia, PA 19101 I'SA

ABSTRACT

This paper describes current work at the University of Pennsylvania centered around providing intelligent help r.:id advice to users of interactive task oriented systems. This work focuses on three genera! themes: (1) Help systems should be active rntla-r t!:an passive; (J) help systems should coniain explicit models of the user, the ta-k and the system utility being u-ed and (3) the help system shoulJ engage in an interactive dialogue with the user in order to identify the information he really needs. An experimental Uv\\, system, WIZARD, has been implemented for the VAX/VMS' operating system to explore some of these issues.

1. Introduction

The state of the art in "help" systems has not changed much in the last dee.vie (or two'). Current help systems typically fall into one of three general categories: key word systems, hierarchical key word systems and network systems, (described more fully in [Finin 8'J]). Present approachs to providing help to inexperienced or infrequent users fails for two important reasons. First, current help systems have no real understanding of who they are trying to help (eg what a partirular user does and does not know, what goals users in general are likely to have), the topic they are providing information about, or the information they can provide Second, current help systems are not interactive. The process of successfully requesting and getting the information the user needs is one in which both the system and the user should be active agents. At any point the party should be able to take the initiative.

As an example of a kind of help that current technology can not provide, consider the problem of the user who doesn't know he needs help. It is quite common for a complex system such as a text editor or operating system to be designed so that a new user can learn a few basic commands which are sufficient to accomplish most tasks Additional commands which greatly extend the convenience and practical power of the system are provided for the user to "grow into". It is common, however, for some users to become trapped by the simple complete set of basic commands and never progress to learning the full power of the system.

A human advisor, watching such a user, might notice that he is not making full use of the system and volunteer advice about using more advanced commands. For example, if you see an inexperienced user issuing the following commands to your operating system:

```
DELETE PARSE PAS.I
DELETE PARSE.PAS.2
DELETE PARSE.PAS.3
```

and leaving the file PARSE.PAS.4, you might offer the following advice:

> It looks like you're trying to delete all of the old versions of the file PARSE.PAS. You can do this directly with the FURGE command. For example, you could have said:
>
> PURGE PARSE.PAS
>
> and it would have had the same effect.

An anoint- sample of the shorlcommings of run cut systems i» lhat somcuiii i..a,. know he needs help but not know how to ask for it Consider ihc plight of a new user on a UNIX system (circa l°7o) lies created hi> first file and would like to have it printed on his terminal. Having had some experience with other operating systems, he make* Mime reasonable giic .–s :it the command name: print, type, list, display, and show None of these work ?o he derides to use i he I'nix heip system to find out what the appropriate comma-.d name is. Again, he tries some familiar (and some unfamiliar) names: help, info, ?, show, teach, learn, describe, apropos, what, please, etc. lie will have to experiment for a long time before he discovers the correct way to get information on the command to print a file on the terminal: man cat

2. A Knowledge Based Approach to Kelp Systems

The ultimate success of an intelligent help system will depend on the depth of its understanding of the task domain, the system it provides information about, and the user's knowledge of the system. Some earlier work has been done on representing task domains (ICienesereth 78] |Ball 80]). representing iinplimentation (|Brarhrnan 78], [Shrager Hi]) and on representing a user's model of a formal system ([Burton 78], |Cioldstein 7'J]). We are attempting to combine these three sources of knowledge into a single, unified knowledge base. We have chosen to u-e the knowledge representation language KL-ONK [Brachman 1978a, PJ78b] as an implementation tool in investigating intelligent help systems.

We view the process of the user asking for help and receiving the information he needs as a complex, interactive process. Both the help system and the user need to play an active role. Each must be able to take the initiative and do some action toward achieving the ultimate goal - the user's receiving the information he needs In particular, we see this process as having at least six identifiable steps, given in the following list For each of the steps we indicate some of the strategies that can be used by the user and the system to satisfy it.

1. Does the user need help? Both the user and the system must know that the user needs help. The user can discover this on his own and communicate it to the system by "pressing the help button". The system might discover it by observing that the user is encountering unexpected errors, by noting the user using a "catalogued bug", or by noting an unexpected inactivity by the user.

2. What information does the user need? Both the system and the user need to establish an initial description of the information that is needed. The user could try to establish this by exploring the information that is available on the system in a top-down manner by asking for very general help and then refining the question. The system could use its model of the user and of the current context he is in to make an educated guess as to what he is most likely to need to know. Another strategy that the system could use is to provide the user with a menu, or a system of menus which enumerate all

the relevant topics.

3. What information can the system -provide? The user and the system need Uwihave some model describing the information that the help .system can deliver. The user could discover this himself if the help system provided a "mcta-help" facility tint, could provide information on the help system itself. The system could try to inform the user about the range of information it could provide by presenting summaries of the available help or by enumerating only what it estimates to be the relevant topics.

J Which offerings match the needs? Once we have a description of the information the user needs and descriptions of the information that the system can provide, we need to identify the best matches In general there will not be a one-to-one match. The user can use his own facilities to compare the descriptions or the system could use some sort of description-matching process to suggest matches.

5. How does one ask for a particular pice- of information? Once the user has derided what information he needs and which of the offered information matches his ne;d, he has to know how to specify that to the help system. A wide range of modes are possible which allow the user to d< scribe the information he v\ -nts: topic names, key words, key words with qualifiers, simple natural language-like phrases and unrestricted natural language. The system, on the other hand, could take the initiative at this step by presenting a menu of likely choices or by selecting the most likely candidate ;md 'imply presenting it to the user

(> Docs the user understand the help? Once the user receives the information he has to understand it. It should be the systems goal to tailor, when possible, the information it presents the user to that user's level of expertise and previous experience The system should be actively trying to verify that the user has correctly understood the information he was given. The user should be able to ask for clarification or examples. The system might be prepared to offer background information (eg. definitions of terms) and alternative presentation of the same information

3. WIZARD

WIZARD is an experimental Help system whose domain is a subset of the VAX/VMS operating system ([Shrager 1081], [Shrager and Finin, 1982]). It attempts to provide inexperienced or infrequent, users with a certain kind of advice to help them learn to use the operating system efficiently.

In developing WIZARD we attempted to focus on the problem of recognizing when the user required help and volunteering advice. Our approach is to recognizes correct yet inefficient command sequences and help the beginner become more proficient by indicating how these tasks may be done more efficiently.

There are several dimensions to inefficiency in operating system interactions: operating systems provide many features (such as wild cards in file names, lists of verb targets, etc) which are meant to minimize the users work (e.g., typing). Consider:

```
SHUNT PROCESS.MEM
SHUNT MEETING.MEM
etc                    a
```

rather than

```
SHUNT *.MEM
```

On another dimension, we measure inefficiency in terms of system resources. The system typically provides special functions which perform operations much more sparingly than more general means would permit Contrast:

```
SCOPY NOTES.* OLDNOTES*
SDELETE NOTES.*.*
```

with

```
SKENAME NOTES* OLDNOTES.'
```

WIZARD recognizes such "less efficient" sequences and constructs help messages that provide either immediate advice or a pointer to a manual or online HELP entry In (lie following example, WIZARD'S advice is shown indented:

```
SCOPY TESTE TEST2.
SHELETE TESTE*
    If you are trying to change the name of tin
    filt TEST I. to TESTti. you could hart done
    tins mart directly with the RUN A Mil
    comma?) <!. (.g.
        $RENAME TEST I. TEST2.
    Typt HEI n RENAME for more information
    on the RENAME command.
```

In our current WIZARD implementation we attempt to provide ;i nmi-interriipt ive interface through the use of multiple windows. WIZARD'* comments are presented in a special pop-up advice window which the user is frc to change the size of. eliminate or simply ignore.

The difficulty of WIZARD'S task is, of course, to recognize when some sequence of commands constitutes a plan that a person is using to achieve a goal We have approached this problem by constructing a catalog of "bad plans" which novice users often use to achieve (ommon goals The problem thus reduces to matching command sequences to descriptions of generic plans from the catalogue. In this application, the matching process is complicated by the following issues.

- Non-contiguity - The individual commands which make up a sequence might be spread out over a session. Each of the intervening commands may or may not affect the goal which the overall sequence is meant to achieve.

- Non-linearity - A single event may play a part in several plans.

- Ambiguity - The mappings from sequences of events to plans and from plans to goals are both many to many. A given sequence may instantiate several plans and a certain plan can be instantiated with a number of sequences. Similarly, a plan might be used to satisfy several goals and a particular goal can be realized by several plans.

> Extensional Knowledge - In general we may need to have detailed knowledge of the user's environment since a given sequence may have side effects or use information

not directly expressed in the syntax of the commands. In order to recognize which of several possible goals is being attempted one may. for example, have to expand "wildcard" patterned filenames and select n.mes from the current directory which are referred to by the command at hand

The goal recognition process is driven by an expectation-based parser [Ricsbeck and SehankJ which uses a knowledge base represented in a network based representation language based on KE-OXE This knowledge base represents many of the concepts for objects and processes which undcrly the task domain, generic and specific commands that the system provides, and the actual history of the interaction between the user and the task system Thus, the mode l of the task and of the system have been merged by having a single taxonomy which contains both generic task concepts (e.g. commands which create files), generic system commands (eg COPY) and individual instantiations of commands issued by a particular user (eg. the COPY command that user TIM issued at 3:02pm on May 3ed) The only information that *WIZARD* keeps about an individual user is which commands he has been observed using and which pieces of advice he has already been given.

When the user issues a operatin system command, individual concepts representing the command are added to the knowledge base If this event matches the initial event in a sequence of a \ lan, a demon is created to monitor the knowledge base for the instantiation of a concept matching the next event in the plan s sequence If a concept matching that event is subsequently instantiated, then the next event in the sequence is monitored for When all of the events in the sequence have occured, the plan is checked to ensure that it satisfies any local and global constraints placed on it (eg what kind of intervening events, if any, can occur).

If all of the constraints are satisfied, then *WIZARD* takes the goal associated with that plan as the users goal.[1] *WIZARD'S* advice is generated from the advice *template* associated with with the plan The advice template may contain references to individual objects (eg file names used as command arguments) in the events matching the plan sequence. Before offering the advice, *WIZARD* checks to see if this user has already been shown it.

4. Conclusions

The *WIZARD* Help system has addressed only ihe problem of recognizing certain situations in which the user may need help or advice. Our current work is directed toward developing better approaches to modeling the individual user (i.e. his interests, level of expertise, etc.), the generic user (i.e. his a priori goals, his planning stategies, etc) and toward making the interaction between the user and help system more of a dialog.

In particular, we have been working on recognizing and responding to a user's misunderstandings |Schuster 83], using a model of the user as a basis for customizing the help offered [Schuster 83] and various spreading activation techniques for identifying relevant help information from a set of weighted keywords |Howe 83],

[1] The current implementation does not handle multiple goals associated with a riven plan.

5. Acknowledgements

This work is partially supported by the National Sei< Foundation under grant number IST8I-12139. Jeff Shr: collaborated on much of this research and is responsible for current *WIZARD* implementation.

6. References

1 Hall, E , and Phil Hayes, "Representation of Task-Specific Knowledge in a Gracefully Interacting User Interface", in Proc AAA1. 1980.

2. Brachrnan, R.. A Structural Paradigm for Representing Knowledge, BBN Report no. 3006. 1978.

3 Brachman, R., E Ciccarclli, N. Grecnfeld and M. Ynnkr, "KLONE Reference Manual". BBN report no. .VS18, 1978.

•I Burton, R and .1 Brown, "An investigation of Computer Coaching for Informal Learning Activities", BBN Report 3911. 1978.

6. Kinin, T.. "Help and Advice in Task Oriented Systems", Report MS-CIS-1982-22, Computer and Info, Science. U. of Penn 1982.

b Genesereth. M , "Automated Consultation for Complex Computer Systems", Ph.D. Thesis, Urpartment of Computer Science, Harvard University, 1978

7 Genescj.-th. M , "The Role of Plans in Automated Consultation". Proc 1.1C M-77, 1977.

8. Howe, A., "HOW? A Customizable, Associative Network Based Help facility". Report MS-OIS-1983-1 1, Computer and Info. Science. U of Perm. I983.

9 Goldstein, I.. "1 he Computer as Coach. An Athletic Paradigm for Intellectual Education", AIM 389, MIT. 1979

10 Riesbeck. C and R. Schank, Comprehension by Computer: Expectation Based Analysis of Sentences in (ontext, Yale University CS report no. 81.

11 Sch;. k, R and R Abelson, Scripts, Plans. Goals and Understanding, Lawerence Erbaum Press, Hillsdale NJ, 1977.

12 Schuster, E and T. Finin, "Understanding Misunderstanding - Recognizing and Responding to User Misunderstandings", report MS-CIS-83-12, Computer and Info Science, U of Penn., 1983.

13 Schuster. *K.*, "Custom-Made Responses: Maintaining and Updating the User Model", report MS-CIS-83-13. CIS, U of Penn., 1983

14. Shrager, .1. and T. Finin, "An Expert System that Volunteers Advice", Proc. AAAI-82, 1982.

15 Shrager, J., "Invoking a Beginner's Aid Processor by Recognizing JCL Goals", report MS-CIS-81-7. Computer and Info. Science, U. of Penn., 1981.

16. Wilensky, R., 'Understanding Goal-based Stories", Yale University Research Report No. 140, 1978.

17. Wilensky, R., "Talking to Unix in English", tecnical report, Univ. of Ca. - Berkeley, 1982.