Task Modeling by the Keywords Extracted from Manual Pages

By

Toshikazu NISHIMURA*, Michihiko MINOH** and Katsuo IKEDA*

(Received March 31, 1995)

Abstract

A task modeling method using keywords related to the functions of commands for intelligent user interfaces is proposed. A task model is a description of functions and operations of the computer.

The task model is described as the name of a task in our method. We define the name of a task as the minimal set of common keywords which definitely distinguish the task. For the name of a task, keywords indicating the concepts of each command are needed. Since the order of the number of the keywords is that of the number of the commands, it is much easier to implement this method than a procedural knowledge based one.

We also propose a method to extract the keywords automatically from the manual pages, the on-line reference manuals for UNIX. Since almost every command is associated with manual pages, the task names can be easily updated when a new command is added to the system.

To show the effectiveness of the task names and the keywords extracted from the manual pages, we direct our attention to users' command histories. We show the effectiveness of these keywords by showing the relationship between the tasks in the history and the task names through statistical analysis.

1. Introduction

This paper describes a method of task modeling for experts using keywords related to the functions of commands for intelligent user interfaces. We also analyze the keywords extracted automatically from manual pages to show that they are suitable for our method.

An intelligent interface is to bridge the gap between a user and a computer and to support dialogues between them. A task model is a description of functions and operations of the computer. This is one of the most important components of intelligent interfaces since all user support, such as plan recognition, automated macro execution, customizing tools, user help and so on, is based on task models.

An intelligent help system, which is to assist users with advice suitable for their goal inferred from their inputs and questions, is a sort of intelligent interface and also has a

* Department of Information Science, Kyoto University, Kyoto 606-01, Japan
** Integrated Media Environment Experimental Laboratory, Kyoto University, Kyoto 606-01, Japan

task model in it.

Since novices lack the knowledge of plans to reach their goals and usage of commands, most intelligent help systems are for novices and the task models in them are described as procedural knowledge bases. For example, UNIX Consultant (UC) is an intelligent help system which can answer users' questions about UNIX file systems in English. The task model in UC is a description of structures of plans for users' goals. Neo-Assist[5] and Command Interface Shell[6], which are intelligent help systems on UNIX, also incorporate descriptions of structures of plans as their task models.

Descriptions of structures of plans are suitable for symbolic reasoning, and are natural and easy to understand[7]. However, they are hard to implement in practical use since the order of the number of plans is that of the number of permutations of commands and there are too many plans to be described by the system designers. Moreover, most systems can not update their task model automatically and can not respond to unknown commands.

Schiele *et al.* propose a method to update a task model by inferring the task structure from the interaction protocols[8]. The method mainly depends on inductive reasoning with some heuristics and can be used for task analysis. However, all the functions of the target commands are needed for this method and the system will not work on unknown commands.

Moriyon *et al.* propose a method to generate help automatically from the model used to implement the interface. Their method is to develop the interface and its help consistently. However, it is not suitable for commands in ordinary computer systems since the number of commands is much larger than that of the interface parts.

Those task models are not suitable for ordinary UNIX computer systems since they will not work for user defined commands and such commands are frequently added to the system. Moreover, task models described as procedural knowledge bases are not suitable to support experts since they have an abundance of knowledge of plans to reach their goals and of usages of commands.

Thus we propose a method to describe a task model for intelligent interfaces for experts. We define a task as a set of commands. In our method, the function of a task is described as the name of the task. We define the name of a task as the set of keywords related to the function of the task which distinguish the task definitely. Though keywords corresponding to each command are described by the system designer, it is easy to implement the method since the order of the number of the keywords is that of the number of commands and the number of commands is rather smaller than the number of permutations of commands.

We also propose a method to extract the keywords automatically from the manual pages, the on-line reference manuals for UNIX. Since almost every command is associated with manual pages, the task names can be easily updated when a new command is added to the system.

To show the effectiveness of the task names and the keywords extracted from the manual pages, we direct our attention to users' command histories. We consider a series of commands in the history to be related to one of the user's goals, so the set of the commands should have a name. Thus we can show the effectiveness of the task names and the keywords by showing the relationship between the tasks in the history and the task names.

Section 2 describes the definition of our task names. We propose a method to extract the keywords automatically from the manual pages in section 3. Section 4 discusses the effectiveness of the task name and the keywords. Finally, we present conclusions and note related future works in section 5.

2. Task Names

We consider a case where a user issues multiple commands to accomplish his task. The task in this situation is described as a tree structure, where the root node represents the user's end goal and other nodes represent sub-goals. Each leaf node in this structure represents a command or a use of a command and contains knowledge related to the command.

Since many experts have an abundance of knowledge of plans to reach their goals and of usages of commands, it is not necessary to support features like the structure of a user's plans, usages of commands and the execution order of commands leading to the goal. So we omit these from the task models and define the description of the situation as the correspondence of the set of executed commands to the name of the set describing the user's goal.

We define a task as a set of commands. We call the number of elements in a task *the size of the task*. We assume each task involves executing two or more commands.

When a user hears the name of a command, he may associate other words. For example, he may associate words like file, print jobs, printer, font and so on for "lpr", the command for printing out files in UNIX. These words are descriptions of the concepts related to the command. The relation between commands and these words is a many-to-many correspondence.

Let's consider the situation where a user issues multiple commands to accomplish a task. The words a user associates with the task are related to the user's plan. These words are the common concept related to the commands, *i.e.* the elements of the task, when we assert that the command combination does not affect the concept.

Thus we can use the concepts related to a user's goal as the common keywords when the keywords indicating the concepts of each command are assigned to the command in advance.

However, the common keywords themselves are not sufficient as the name of a task since some tasks have identical common keywords. For example, when the set of common keywords for a task T is K, all subsets of T have K as the set of common keywords.

Thus we define the name of a task as the minimal set of common keywords which distinguishes the task definitely. When there is no such set for a certain task, the task does not have a name. A different name is given to a different task.

All we must prepare for this method are the keywords for the concepts of commands. Since the order of the number of keywords is that of the number of commands, it is much easier to implement this method in practical use than to implement the task models as procedural knowledge bases.

Our method for the task model is formally defined as follows:

Assume that the number of commands is finite. U_C denotes the set of all commands and 2^{U_c} denotes the power set of U_C. Keywords are finite-length character strings. We assume that the set of all keywords U_K is a finite set. 2^{U_k} denotes the power set of U_K.

f': $U_K \rightarrow 2^{U_c}$ denotes the mapping of the set of keywords onto the power set of U_C. $f'(k)$ is the set of commands which are assigned a keyword k, provided that $\forall k \ni U_K$, $|f'(k)| \geq 2$.

f: $2^{U_k} \rightarrow 2^{U_c}$ denotes the mapping of the power set of keywords onto the power set of U_C which satisfies the following conditions:

1. $f(\phi) = U_C$.
2. $f(\{k\}) = f'(k)$.
3. $f(K) = f(K_1) \cap f(K_2)$, where $K \subseteq U_K$ and $K = K_1 \cup K_2$.

Definition: $t(C)$ which satisfies the following conditions is the name of task C.

1. $t(C) \subset U_K$.
2. $f(t(C)) = C$.
3. $f(S) \supset C$ for all $S \subset t(C)$.

3. Extraction of Keywords from Manual Pages

One of the most important issues of our task modeling is the keywords related to the function of the task. One way to construct the keywords and the mappings is that the system designer describes them. However, the system designer must update the keywords and the mappings in this method when a new command is added and the system is extended. Instead, we attempted to extract the keywords from the manual page of the reference manuals of the UNIX system automatically when the system is extended.

The method to extract the keywords is as follows.

1. Remove comments and heads in manual pages and cut them down into words.
2. Restore conjugations to bare infinitives and derivatives (inflections, declension) to underivate. Remove articles, auxiliary verbs and conjunctions since they are not specific to the commands.
3. Remove words which appear only once on each page since some manual pages have references related to other commands.
4. Remove words which do not appear in the manual pages of the other commands since these words can not be used for a task name. The remaining words are the keywords.

Table 1 shows examples of the keywords extracted for our experiment described later. These keywords seem appropriate as associations from the commands.

Table 1. The examples of keywords

Command	Keywords
vi	character classification command contain edit editor environ ex file flag input option set tag test text variable vi view
lpr	character contain copy data example file font job name option page print produce send specify spool standard use

4. Keywords Analysis

It is important to evaluate the keywords since they are automatically generated. This section describes a method for evaluating the keywords.

In our method, the tasks that can not be definitely distinguished by any set of keywords have no name. Since the number of names is very small compared with the number of tasks as shown later, random tasks should not have a name while the tasks for the user's goal should have a name.

The quantitative analysis of the keywords by the system designer is very hard since the order of the number of tasks is about n to the power m, where n is the number of commands and m is the maximum size of a task. Though the evaluation of the application using our task model may show the effectiveness of the keywords, it is difficult to evaluate the keywords independently.

Thus we utilize sequences of commands in the history for the evaluation of the keywords. Most sequences of commands aim to accomplish the user's goals. The keywords are effective for our task model when most of the tasks in the history have a name, while not effective otherwise.

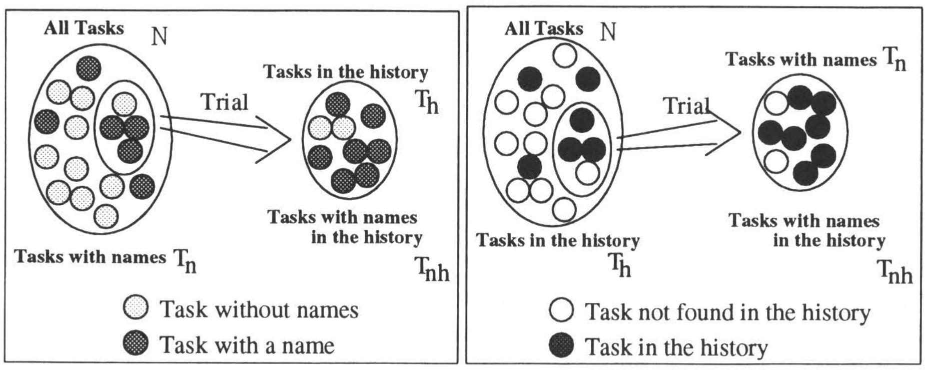
We call the command set which consists of sequences of commands in the history *a task in the history*. We discuss the effectiveness of the keywords by showing the relationship between the tasks in the history and the existence of a name for tasks.

4.1 Statistical Analysis

Let N denote the number of tasks. $N = n_c\, Cn_t$, where n_c is the number of commands, n_t is the size of a task, and C denotes the combination function. Let T_n denote the number of tasks with names whose size is n_t. Let T_h denote the number of tasks in the history of size n_t. Let T_{nh} denote the number of tasks in the history with names whose size is n_t.

When $T_n = T_h = T_{nh}$, the tasks in the history and tasks with names are identical. However, $T_n = T_h = T_{nh}$ is not always true since all sequences of commands in the history are not for the same tasks. Thus we will show the relationship between the tasks in the history and tasks with names by a statistical analysis.

Let us consider the following two trials A and B and set up a null hypothesis H_0.


All Tasks N
Tasks in the history T_h
Trial
Tasks with names in the history T_{nh}
Tasks with names T_n
Task without names
Task with a name
All Tasks N
Tasks with names T_n
Trial
Tasks in the history T_h
Tasks with names in the history T_{nh}
Task not found in the history
Task in the history

(a) Trial A (b) Trial B

Fig. 1. The concepts of the trials A and B

Trial A The number of tasks is N. Among these, T_n tasks have names. Among the T_h tasks in the history, T_{nh} tasks have names.

Trial B The number of all tasks is N. Among these, T_n tasks are in the history. Among T_n tasks with names, T_{nh} tasks are in the history.

Null hypothesis $\mathbf{H_0}$ The results in Trial A and B are coincidental. There is no relation between a task found in the history and the existence of a name for that task.

Figure 1 shows the concepts of the trials A and B. If H_0 is statistically rejected, it shows the relationship between the tasks in the history and tasks with names.

The probability p that a random task will have a name is T_n/N in trial A based on H_0. Thus the expectation m of the number of tasks with names and its standard deviation σ in trial A are as follows:

$$m = T_h \times p = T_n T_h / N.$$

$$\sigma = \sqrt{T_h \cdot \mathrm{p}(1-p)} = \sqrt{T_h T_n (N - T_n)}/N.$$

The m and σ in trial B are as follows:

$$m = T_n T_h / N.$$

$$\sigma = \sqrt{T_h T_n (N - T_h)}/N.$$

Since the distribution of the probability variable T_{nh} can be regarded as a normal distribution $N(m, \sigma)$, the distribution of the probability variable T' is $N(0, 1)$ where $T' = (T_{nh} - m)/\sigma$.

4.2 Experiment and Discussion

We have collected about 400,000 lines of history from dialogues between 18 colleagues in our laboratory and a computer. We conducted experiments on the commands which were used at least twice and which have manual pages. There were 280 such commands on these commands. First, we extracted the keywords from the manual pages of these commands. The number of keywords was 1280. Second, we extracted the tasks in the history and the tasks with names. Table 2 shows the number of these tasks. The maximum size of a task is limited to 4 because T_{nh} was 0 when the size of tasks was over 4.

The number of tasks with names is much smaller than that of all tasks in table 2. This implies that the task names are rare. Though some of the tasks in the history have names, the number of tasks with names and the number of tasks in the history are quite different. Since it is not clear from table 2 whether the keywords extracted automatically from manual pages are effective or not, statistical analysis is needed.

Table 2. The number of tasks

Task size	The number of tasks	T_h	T_n	T_{nh}
2	39060 ($_{280}C_2$)	2660	4118	513
3	3619560 ($_{280}C_3$)	5806	19512	98
4	250654530 ($_{280}C_4$)	6712	52811	4
Sum	254313150	15178	76441	614

T_h : The number of tasks in the history
T_n : The number of task nmaes
T_{nh}: The number of tasks with names in the history

Table 3. The results of Trials A, B

Task size	Expectation m	T_{nh}	Trial A		Trial B	
			σ	T'	σ	T'
2	280.4	513	15.87	14.65	16.20	14.56
3	31.50	98	5.579	11.95	5.590	11.93
4	1.414	4	1.189	2.174	1.189	2.174

$T'=(T_{nh}-m)/\sigma$

Table 3 shows expectation m, standard deviation σ and probability variable T'. The null hypothesis H_0 is rejected with very small risk when the task size is 2 or 3. This shows the relationship between the tasks in the history and the task names. When the task size is 4, the null hypothesis H_0 is also rejected with about 3.00% risk, which also indicates the relationship between the tasks in history and the existence of a name for tasks. Thus we conclude that the task names with the keywords extracted automatically from the manual pages are effective regarding the tasks in the history.

5. Conclusion

We proposed a task modeling method using names of tasks with keywords related to the functions of commands for intelligent user interfaces for experts. The name of a task is a set of keywords which distinguish the task definitely.

We extracted the keywords automatically from manual pages for system expansion. We also show the effectiveness of these keywords by showing the relationship between the tasks in the history and the task names through statistical analysis.

The keywords for the task names are related to the functions of commands. Therefore command clustering with the keywords may also improve the help messages of commands in traditional intelligent user interfaces.

Since our method mainly depends on keywords, it can not be applied to pointing operations with a mouse employed in many graphical user interfaces (GUIs). Applying our method to GUIs is a subject for future study.

Acknowledgment

We would like to thank Mr. Tatsuya SUGAHARA who implemented the method to extract the keywords described in section 3.

References

1) Norman, D. A. and Draper, S. W.: "User Centered System Design," Lawrence Erlbaum Associates (1986).
2) Hancock, P. A. and Chignell, M. H.: "INTELLIGENT INTERFACES Theory, Research and Design," Elsevier Science Publishers (1989).
3) Wilemsky, R., Arens, Y. and Chin, D.: "Talking to UNIX in English: An Overview of UC," Comm. ACM, **27**, 6, pp. 574–593 (1984).
4) Chin, D.: "A Case Study of Knowledge Representation in UC," Proc. 8th Inter. Joint Conference on Artificial Intelligence, pp. 388–390 (1983).
5) Uehara, T., Uehara, K. and Toyoda, J.: "Simulating User's Consideration for Inferring User's Intention," JSAI SIG-HICG-8902-2 (1989).
6) Aoe, J., Maeda, A., Kujime, H. and Morimoto, K.: "Understanding of Command Using Natural Language Inputs—an Aid to UNIX Systems—," IEICE NLC90-44 (1990).
7) Parsaye, K. and Chignell, M. H.: "Expert systems for experts," Wiley (1988).
8) Schiele, F. and Hoppe, H. U.: "Inferring Task Structures from Interaction Protocols," INTERACT'90, pp. 567–572 (1990).
9) Moriyon, R., Szekely, P. and Neches, R.: "Automatic Generation of Help from Interface Design Models," CHI '94 Conference Proceedings, pp. 225–231 (1994).
10) Sequent Computer Systems: "DYNIX Programmer's Manual" (1987).