Winter 1999

Mentor: Intelligent Tutoring Systems That Learn When to Help

R. David Sigsby
Grand Valley State University

Follow this and additional works at: http://scholarworks.gvsu.edu/mcnair

Recommended Citation
Mentor: Intelligent Tutoring Systems That Learn When to Help

R. David Sigsby
McNair Scholar

Roger Ferguson, Ph.D.
Faculty Mentor

James Beale, Ph.D.
Faculty Mentor

ABSTRACT:
Frustration is a problem common to all people learning a new technology, particularly computing technology. Intelligent Tutoring Systems (I.T.S.) can offer help based on a user's goal or ability. What these systems fail to take into account is that help is useless if the user is not ready to accept it. This research has two main aspects: the content of help and the timing of that help. Thus the goal of this research is to develop a program, called Mentor, which first offers good helpful direction to the user and second, learns the appropriate time to intervene. Mentor is being developed in modular form, the modules are as follows: (1) observing the user's behavior within a software environment, (2) statistical analysis of the users behavior, (3) a set of heuristics to decide what help to offer and when, (4) a means of presenting help to the user, and (5) a feedback system to measure user responses to the help offered.

Introduction
A problem common to all people learning a new technology is that of help, particularly for the novice. Users often do not realize they need help, don't know where to get help, and have difficulties understanding help when they get it. When a user decides to look for help, current resources available to them include manuals (both on-line and printed), technical help from more advanced users, and help from paid trainers. Qualified trainers are in high demand and have become expensive. With a large group of people needing to be taught and a comparatively small group of trainers, it is increasingly difficult for trainers and advanced users to provide adequate instruction to novice users. This is frustrating for the novice seeking help. Being limited by what information the author chooses to at the time of publication, help manuals and textbooks cannot provide the same quality of help as trainers. As a result, such help is often confusing and intimidating for the novice user and trivial for the advanced user, providing no consideration for varying styles among individuals. Also, such written forms of help quickly become outdated, leading to wrong information and high levels of frustration for the user. Clearly, providing inexpensive and effective help to computer users is a problem that needs to be addressed.

The ideal solution to this problem is a system which would observe, assess, and aid the users learning process, i.e., "systems that are capable of providing the quality of support of trainers but are as cheap and accessible as manuals" [Mallen]. Intelligent tutoring systems such as GUIDON (1983), WEST (1982), LISP TUTOR (1985), and BUGGY (1978) [Smith] have been in place for some time and have not met the needs of users. The existing systems are overdue for an advancement: "to be able to gather information on how people think about problems and present solutions modeled on that." [Nass] By offering help tailored to individual needs, we can increase the effectiveness of available help systems.

Our objective is a software program for use as a personal tutor in introduction to computer programming courses. In other words, the goal is to develop a program which monitors and evaluates a student, offering effective help when it is needed. This can provide the individualized help needed without the student having to ask, possibly even before the student realizes help is needed. Though introduction to computer programming students are the current subjects of research, such a pro-active tutoring program can be extended to various applications, e.g., word processing, databases, spreadsheets, internet help, and so on.

Methods
Mentor is being developed for use by introduction to computer science students (CS 162 at Grand Valley State University) using the JAVA programming language. The students use of the Integrated Developers Environment (IDE) is monitored by capturing keystrokes via Stealth Logger Core [Birjukov] (a commercially available monitoring tool).

Mentor is being developed in modular form. The modules are to observe, analyze, decide, offer help, and measure response (see Figure 1). These phases build a loop in which Mentor is constantly evaluating the user.

```
Observe

Analyze

Decide

Offer Help

Measure Response
```
For the observation module, student behavior is recorded. For example, when
the user saves, Mentor records the 'control' key being pressed followed by the 's' key
being pressed (the keyboard equivalent for saving). Behavioral variables consist of
actions taken by the user and currently include saves, deletes, runs, compiles, setting
of watches, and setting of breakpoints. When the user performs one of the above
actions, Mentor records the time of the action as well as the number of characters
since the action was performed last. Mentor will also be recording the number of
characters a user types per minute.

The analysis module gathers information about the user through statistical
analysis of the recorded behaviors. Mean and standard deviation are figured for
each behavioral variable. Initial work involves looking at a sample population of
students to identify statistical differences between novice and advanced users
through the observed behavior. For example, it is expected that an advanced user
will set more watches more often than a novice user. These differences will be used
to decide the content of the help.

The decision module weighs the user's current behavior versus that of the
advanced user. Currently Mentor offers help at set intervals based solely on the
ability of the user. For example, Mentor alerts the user to save when the novice
user deviates from the behavior of the advanced user. The user will then evaluate
how helpful Mentor is. For future research, Mentor will be allowed to tailor
the timing of its help to the individual behavior. In other words, in order to offer
help at a time when the user will be receptive, Mentor may decide to wait to
offer help even though it is observed that the user needs it. The users will then re-
evaluate Mentor's effectiveness (see experimental procedure below).

The module that offers the help prints a message to the user in a window
that belongs to Mentor. Examples of the ideas expressed in the help include: save
your work, slow down, take a break, use watches, use breakpoints, compile your
code, start debugging.

The module for measuring the user response has several possible methods of
implementation. The decision for final implementation of this module relies
heavily on the observed behavior of the users, a phase which has not been com-
pleted. Methods being considered for measuring responses are a button which
the user clicks to acknowledge the message or for Mentor to observe if the user
has followed the advice. An example of the latter is that when Mentor tells the
user to save their work, Mentor will be able to observe if they do indeed save
within a certain amount of time. How often the user responds favorably to
advice would then be used by Mentor to determine when help will be offered in
the future.

Experimental Procedure
In measuring whether or not a user responds to help, it is essential to look at
two factors: is the help appropriate to what the user needs and is it offered at
the correct time. These two factors are to be tested in two separate experiments.

The first experiment is to find content for the help messages which is bene-
ficial to the user. The sample will be divided into a control group and an
experimental group. Classroom instruction, the textbook, the professor, and pre
and post tests are conditions that will remain constant between the two groups.
Pre and post test measures will be directed at identifying levels of satisfaction and
frustration with the class overall, the professor, assignments and projects, the IDE,
and the programming language. As the contents of the tests are not yet complete,
other issues may be addressed. The experimental group receives help from Mentor
while the control group does not. This allows us to determine how useful the
help is by factoring out effects such as a student's frustration with the professor,
problems with the textbooks, etc. By offering help that is useful, we reduce the
chance of the user not responding due to the help being too simple or too compi-
cated. This experiment may be repeated until it is determined that the content of
the help is valuable to the user.

The second experiment addresses the issue of timing. The sample again consists
of a control group and an experimental group with the same conditions common
to both as in the first experiment. In addition, both groups will receive help from
Mentor. The control group will receive help at set intervals, i.e. if the user has not
saved after a set period of time they will be prompted to save their work. In the
experimental group, Mentor will be allowed to tailor its timing to when the
user will be receptive. For this group, initial help messages will be offered with the
same timing as the control group. As Mentor observes the users rate of
response to the help, the timing will be tailored to when that user is more likely
to respond. In other words, based on results from the first experiment Mentor
may observe that the user needs help but decides the user will not be responsive at
this time. Mentor would then wait to offer the help at a more effective time.

The expected results of the second experiment are that the post tests of the
experimental group will show higher levels of satisfaction and lower levels of frus-
tration than the post tests of the control group. This would show that timing
makes a difference in a users responsiveness to help. More importantly, the
expected results will demonstrate that Mentor is able to teach itself how to alter
the timing of its help messages to make them more effective.

Future Work
We begin by expanding Mentor's capabilities. While the immediate application of
Mentor is for computer programmers using the Visual Café environment, its ultimate goal is for general application. The first step is to expand the means of observing the user. We begin looking at more detailed behavior, allowing Mentor to be more accurate in the timing and content of help, i.e., to offer effective help when the user is most receptive. Mentor begins to concentrate more on building a model of the users patterns of behavior by looking outside the use of the IDE. One way this can be done is to observe use of the mouse and the operating system as well as asking the user or allowing the user to “talk” to Mentor directly. It is also possible to use a camera to observe the users movements, though that would be beyond the scope of the research group at this time. The idea is to begin gathering information about the user and build a model of the user based on behavior, with the focus being on finding the best time to interrupt the user and offer them help.

References

Birjukov, Andrei. Stealth Logger Core, copyright Stealth Software, http://ssoft.virtualave.net. Email to: estla@kodu.ee.

