Writing Center Journal

Volume 8 | Issue 1 Article 3

1-1-1987

Getting Smart with Computers: Computer-Aided Heuristics for Student Writers

Fred Kemp

Recommended Citation
Kemp, Fred (1987) "Getting Smart with Computers: Computer-Aided Heuristics for Student Writers," *Writing Center Journal*: Vol. 8 : Iss. 1, Article 3.
DOI: https://doi.org/10.7771/2832-9414.1147

Getting Smart with Computers: Computer-Aided Heuristics for Student Writers

Fred Kemp

ERIC and NCTE combined, in 1983, to produce a small, fifty-page booklet giving English teachers the no-nonsense lowdown on the use of computers for instruction. Its name was straightforward, *Computers in the English Classroom*, and its advice was traditional. After mentioning various drill and practice and record-keeping possibilities, the document informed us, with time-honored NCTE gentleness, that "the value of the computer lies in the fact that it provides one more tool for the teacher to use." It then made what seems to me a manifesto of sorts. "[The computer] frees the teacher from certain mundane chores so that instructional time is better utilized."

Isn't this the way most of us have always thought about computers, as mechanical servants which can take over "certain mundane chores" so that we can get to the higher-level stuff? When you think about it, the idea is not all that comforting. It lies at the heart of the scary theory that computers intended to replicate low-level skills may someday co-opt skills considerably above the "mundane-chores" category so that the servant becomes the master or, at the very least, the master finds himself tailoring and limiting his activities for the convenience of the servant.

The concept of a "servant-master" relationship between computer and human being suggests an anthropomorphic view of computers which, I think, channels our attitudes and severely limits our options in using computers. What I call the "Replacement Fallacy," the belief that computers are most successful when they are most human, hems us in between

two opposite but equally prejudicial theories: (1) either computers can't really replace human beings, and we are fools to depend on them, or (2) they *can* replace human beings, and we professionals are in danger of being superseded by machines. Both positions are naive. The criticisms of computer-assisted instruction that English faculty often engage in arise more from such stereotypical assumptions than from any inherent limitations in the machines and software.

Let's assume, for the sake of argument, that computers are not intended to replace human decision-making, to duplicate any level of mental skill, but are intended instead to extend human understanding, much as telescopes extend human vision. No one accuses the telescope or the microscope of trying to replace eyesight. Telescopes examine stars and microscopes examine microbes, and I believe that the real value of the computer in education is not that it can replace teaching chores or teachers, or even teaching assistants, but that it can model and examine heretofore invisible aspects of how people learn, decide, and express their decisions. The effective instructional methods that result will probably have only a distant relationship to methods previously employed by teachers. We need the imagination to experience what may be entirely new perspectives and the courage to test them sincerely.

Probably the first and best explanation of a non-robotics view of computers in education is Seymour Papert's *Mindstorms*, which appeared in 1980. Papert is best known for developing *LOGO*, a simple computer programming language centered upon "turtle geometry" and intended not for manipulating computer activity, as *BASIC* and Pascal are, but for teaching children procedural methods of solving problems. I say "teaching," but *LOGO* depends very little on instruction in the traditional sense. The student (primary-school age and even younger) "commands" the movement of a design-making arrowhead (called the "turtle") on the video screen by entering simple directions in English. In order to produce more and more complicated designs and pictures, the student has to employ more and more commands in structures of greater complexity, usually involving loops, subroutines, and recursion. The student is not directly told the power of integrated procedures, but internalizes the methods naturally as she goes about making designs. Any counterproductive command or sloppy thinking manifests itself immediately on the screen, and correction is easy and forgiving. As Papert says, "In teaching the computer how to think, children embark on an exploration about how they themselves think. The experience can be heady: Thinking about thinking turns the child into an epistemologist, an experience not even shared by most adults" (19).

LOGO deserves a much more detailed examination than I can give it here. But two things should be clear. The first is that the computer language

https://docs.lib.purdue.edu/wcj/vol8/iss1/3
DOI: 10.7771/2832-9414.1147

strongly encourages coherent and well-structured thinking, the sense of problem-solving as a self-conscious process, and the value of making corrections without fear. The implications for both mathematics and composition seem obvious. The second thing that should be clear is that *LOGO* is not simply a pre-computer teaching method grafted onto the computer, but rather a new kind of instruction arising from characteristically computer, as opposed to human, abilities.

Ironically, *LOGO* engenders a much greater sense of mastery over machine than drill and practice tutorials and other transplanted methods. When computer abilities are confused with human abilities, the computer comes off as either a dud or a villain, and the user feels frustrated or even used (by the instructor, the program, or the computer itself). But when computer activities are accepted on their own terms, even by five-year-olds, the result is usually the kind of excitement that goes hand in hand with a sense of achievement and newly experienced authority.

Most computer-assisted writing instruction is steeped in the Replacement Fallacy, and that is why much of the programming seems to be (after the initial strangeness of the computer has worn off) ineffectual and even annoying to the user. The instructors who are buying the software are demanding a kind of performance from the computer which is inherently self-limiting. To hear most of them talk, the ultimate program would be one that grades papers. Failing to achieve that capacity, they simply lower the level of human skill they would automate.

The trouble with duplicating any instructor skills on the microcomputer in any non-trivial sense lies in the computer's inability to perform Natural Language Processing ("NLP" to the *cognoscenti*). Natural Language Processing is the oft mentioned "holy grail" of Artificial Intelligence research, and can be broadly defined as the ability of the computer to understand and respond to normal discourse. In simplest terms, computers cannot "understand" words or phrases that fall outside very restricted lexicons. The instructions by which even highly sophisticated programs operate are quite specific and must be learned by the user. Since these "command languages" are very much different from the way you and I would normally communicate our wishes, users are often frustrated and must depend on manuals to supply this human-computer "interlingua."

The point is, without Natural Language Processing, computers can't read in any real sense. The only kind of evaluation of a text they can provide is evaluation based on two processes: quantification and matching. A program can, for instance, provide a total word count by counting the spaces between words. Or it can discover spelling errors by matching every word in the text to entries in a dictionary. Using one or the other or both of these techniques,

a computer can also give an average word count per paragraph, a graph describing relative sentence and paragraph lengths, the relative instances of active and passive verbs, and can isolate potential trouble spots, such as troublesome homonyms, hackneyed phrases, gender-biased terminology, and the most obvious syntax errors. The more powerful the program (and computer), the more ingenious the forms such "style checking" takes. Vocabulary and syntax levels, for instance, can be obtained by combining in various formulas the number of letters per word, the number of words per sentence, and (by isolating coordinators and subordinators) the apparent complexity of the sentence structure. This is a sample of the type of style-checking found in microcomputer programs like *HBJ Writer* and *HOMER*, and in the more powerful programs like *Writer's Workbench* and *EPISTLE*.

You may marvel, and many do, that computers can discover and describe in minute detail these characteristics of a student's paper, but I have a different reaction. Taking my cue from Artificial Intelligence research, I look for truths about human discourse from what the computer *can't* do. The fact that a computer can perform the above analyses in ready fashion while being completely locked out of the content of a paper suggests to me that computer editing, while no doubt useful commercially and in discourse analysis, is not very useful pedagogically. Style-checkers promote a band-aid approach to writing instruction and direct time and attention away from the larger issue: the quality of the student's ideas and the relationship between those ideas and the form of expression she employs.

I am not saying that variety in sentence length is unimportant or that inordinate use of passives does not weaken writing. Depending upon the demands of the content, all the style-checking functions described above may have considerable bearing upon the polish and effect of a paper. The problem lies in removing the least sophisticated elements of style (those which can be discovered by quantification or identical matching) from an ancillary position in evaluation and, as a computer must, giving them predominance. Influenced as we are by the Replacement Fallacy, we try to substitute computer skills for human skills, but the only evaluation skills the computer now possesses are those which ignore content completely. For the student (and maybe the instructor) who depends upon such programming, the important emphasis in writing is skewed and the writing act trivialized.

Word processing is a truer appropriation of intrinsic computer characteristics, though an unimaginative one. The idea that a word processer, however, is simply an improved typewriter is foolish. Word processing adds an entirely new stage to the production of text, one in which an extended text is as fixed in the computer's memory as it would be typed on a

https://docs.lib.purdue.edu/wcj/vol8/iss1/3
DOI: 10.7771/2832-9414.1147

sheet of paper, and yet at the same time almost as fluid and amenable to revision as it is when it exists only in the writer's mind. It's the writer's equivalent to having his cake and eating it too. The old dichotomy between writing as process and writing as product largely disappears, because the "paper" no longer exists as the final, set-in-concrete printout, but rather as the disk file which exists in a thoroughly malleable form. Word processing puts all our exhortations about revision and multiple drafting into a mechanism that makes revision so easy and natural that the concept of separate drafts itself begins to disappear. We now see more clearly than ever that the wrong-headed commitment to writing as product most students display is not because they aren't listening, but because the handwritten or typed essay format penalizes them severely for attempting otherwise.

But as attractive and useful a device as word processing is, it does not represent a particularly effective pedagogical implementation of the computer. An English department microlab given over entirely to word processing is no doubt assisting its students to write better, but it is also squandering its computing potential. When, on the other hand, the computer is employed as a specifically heuristic device, it can demonstrate to both instructors and students aspects of the interrelation of creativity and expression which, without computers, would remain invisible.

Let me explain this by returning briefly to *LOGO*. The beauty of *LOGO* is that the student acquires problem-solving techniques without ever having those techniques abstracted and explained. *LOGO* doesn't talk *at* a student the way drill-and-practice tutorials or even style-checkers do. Instead, the student is provided a goal and a set of clear-cut procedures for achieving that goal. It is up to the student to command those procedures in such a way that the goal is achieved; skill at manipulating procedures in ever more complex structures is obtained largely by trial and error. The product (the design or picture) is criticized only by the student himself, not by the computer. The same technique can be applied to writing instruction.

We at the University of Texas English Department Computer Research Lab are working with programs that employ a *LOGO*-type heuristics method. These programs are variously called "dialogue programs" or "interactive questionnaires," but such bland descriptors give little hint of the potential power of the software design. Hugh Burns, who is currently head of the Intelligent Systems Branch of the Air Force's Human Resources Laboratory in San Antonio, pioneered the use of this type of program in 1979 while working on his doctorate at the University of Texas. In his *BASIC* program *Topoi*, Burns reworked Aristotle's twenty-eight enthymeme topics (see Aristotle's *Rhetoric*) into a series of questions which probe the student's understanding of his own thesis. A complete discussion of *Topoi* and two related programs, *Burke* and *Tagi*, can be found in Burns'

dissertation, *Stimulating Rhetorical Invention in English Composition Through Computer-Assisted Instruction* (1979), and in a summary article under the same title written with George Culp (*Educational Technology*, Aug, 1980).

In essence, *Topoi* asks the student to defend his understanding of his subject in response to specific questions concerning definitions, relationships, circumstances, comparisons, and testimonies. Obviously, open-ended questions do not combine to form strict procedural algorithms as the turtle commands do in Papert's *LOGO*, but there is considerable methodological similarity between the two processes. The *Topoi* user has a goal: to complete a writing assignment, which usually means to orchestrate a thoroughly investigated explanation or argument. A thorough investigation is procedural; there are steps that an expert writer uses to insure that her audience does not get lost in gaps or get blocked by inconsistencies. *Topoi* provides the student these steps interactively, as questions or prompts which provoke specific responses concerning specific subjects. By way of comparison, Elizabeth Cowan, in her composition textbook *Writing* (1983), introduces the topoi ("cubing") the only way a textbook can, as one or more abstracted methods for generating ideas (21). Textbooks necessarily employ what Papert criticizes as an unproductive instructional situation, one in which the student is "in the position of listening to explanations" (20). In Burns' *Topoi*, the student is never told that his subject must be well defined; he is simply asked to define it. Eventually the self-interrogative procedures which make up *Topoi* will be internalized, as will the necessity of self-interrogation itself, but that lesson is secondary as far as the student is concerned. To the student, the first job is getting the paper written and handed in. Just as the commands in *LOGO* force immediate results on the screen, the prompts in *Topoi* produce immediate text. Students who would merely skim the tenets of Aristotelian invention as presented in a book will actively employ the very same tenets in *Topoi*, because the results are real and directly applicable to immediate real-world requirements.

As innovative as Hugh Burns' three programs were, they were written for a mainframe computer and perform somewhat awkwardly on the microcomputer. In an article anthologized in William Wresch's *The Computer in Composition Instruction: A Writer's Tool* (1984) and titled "Recollections of First-Generation Computer-Assisted Prewriting," Burns describes some of the deficiencies of his original programs, but he remains convinced of the power of open-ended programming. Others interested in computer heuristics have employed similar techniques in their own programs, most notably Valerie Arms (*Create* and *Recreate*), Helen Schwartz (*SEEN*), Ray Rodrigues (various experimental programs), and, recently, William Wresch (*Writer's Helper*). The tendency has been, however, to include the interactive

https://docs.lib.purdue.edu/wcj/vol8/iss1/3
DOI: 10.7771/2832-9414.1147

questionnaire as simply one of several heuristic devices in a program, as in *Writer's Helper*, or as an aspect of a larger emphasis, as in *SEEN*.

As part of our ongoing "Project Invention Heuristics," the Computer Research Lab is exploring Burns' original design in a variety of formats, principally under the software title *Idealog*. Burns himself has worked closely with us, and he and U.T. English Professor Jerome Bump, director of the Computer Research Lab, co-taught an English department graduate course in the fall of 1986 entitled, "Research in Rhetoric: English and Computers." For well over a year we have been testing the various versions of *Idealog* on selected groups of students, less for definitive judgments about the software's effectiveness than as part of its ongoing development. This development is described in greater detail in my recent article in CCC, "The User-Friendly Fallacy in Computer-Assisted Writing Instruction" (Winter, 1987).

Those in Artificial Intelligence research or those interested in cognitive psychology may complain that I stress the difference between human beings and computers too much, that the entire concept of computer cognition modeling (upon which the heuristics software I have described above is based) depends on the assumption that human and computer mental processes are actually similar, differing mainly in the degree of complexity. I would agree, but stress that the difference in the degree of complexity is so vast that, especially in terms of Natural Language Processing, the similarities are theoretical, not practical. Such notables in Artificial Intelligence as Roger Schank of Yale and Terry Winograd of MIT dismiss the possibility that NLP will enter real-world computing anytime in the foreseeable future (see Schank's *The Cognitive Computer* and Winograd's "Computer Software for Working with Language").

In the meantime, computers can do marvelous things for us in our classrooms and learning labs, but only if we are imaginative enough to forsake the anthropomorphic prejudices of robotry and develop truly innovative instruction based upon characteristically computer abilities. Ironically, this development will depend much more upon the insights of the experienced writing instructor than those of the trained computer scientist. Just as the cognitivists working with the Natural Language Processing are having to transcend stereotypical and traditional concepts of what language is and how language works, those who would tap into the inherent power of the computer in writing instruction are going to have to employ a very sophisticated, and possibly new, understanding of the writing process. It may turn out that we in the humanities gain the most from the computer revolution.

Works Cited

Burns, Hugh. "Recollections of First-Generation Computer-Assisted Prewriting." *The Computer in Composition Instruction: A Writer's Tool*. Ed. William Wresch. Urbana: NCTE, 1984. 15-33.

_____. "Stimulating Invention in English Composition through Computer-Assisted Instruction." Diss. U of Texas, 1979.

Burns, Hugh, and George H. Culp. "Stimulating Invention in English Composition through Computer-Assisted Instruction." *Educational Technology* 20.8 (1980): 5-10.

Papert, Seymour. *Mindstorms: Children, Computers, and Powerful Ideas*. New York: Basic, 1980.

Schank, Roger, with Peter Childers. *The Cognitive Computer: On Language, Learning, and Artificial Intelligence*. Reading: Addison-Wesley, 1984.

Standiford, Sally N., Kathleen Jaycox, and Anne Auten. *Computers in the English Classroom*. Urbana: ERIC & NCTE, 1983.

Winograd, Terry. "Computer Software for Working with Language." *Language, Writing, and the Computer: Reading from Scientific American*. New York: Freeman, 1984. 61-72.

Wresch, William, ed. *The Computer in Composition Instruction: A Writer's Tool*. Urbana: NCTE, 1984.

Fred Kemp is Associate Director of the University of Texas English Department Computer Research Lab and a doctoral student in English. He is the author of several articles on computer-assisted writing instruction. He has also written instructional software, including the computer program *Idealog*, and is currently completing his dissertation, "The Theory and Implementation of Computer-Modeled Invention Heuristics in Composition."