RESEARCH IN WORD PROCESSING NEWSLETTER

Dr. Bradford A. Morgan and Dr. James M. Schwartz, Editors
South Dakota School of Mines and Technology Rapid City, SD 57701 Phone: (605) 394-2481

Vol. 3 No. 3 March 1985

Micros, Minis, and Writing: A Critical Survey

Richard Elias

Judging from the diversity of projects now underway, it appears that every college and university is moving the computer into the writing class. In fact, I know of a college where the dean told the director of writing to begin using computers. "For what?" the director asked, "For word processing? For CAI?" "I don't care," the dean said, "just start using them." But there are serious problems in the projects currently underway, and there is good reason to look beyond the current craze to a more stable future.

At present most programs are microcomputer-based. Consequently, they are severely limited as microcomputers do not have the power, speed, or storage required for anything very sophisticated. This may change as micros grow more powerful and minicomputers get smaller. Over the next few years we can expect that the distinction will gradually blur and eventually disappear.

The inevitable technological convergence suggests that it is foolish for schools to invest heavily in microcomputer software development, at least not for the current generation of microcomputers. Instead, one strategy is to develop software that uses the capabilities of current-generation minis that next-generation micros will probably have. The limitations of currently-available educational software will force educators to re-evaluate the usefulness of computers in the classroom. They will not reject computers. Instead, they will reconceptualize the computer's value as a teaching tool and will concentrate on developing educational software and appropriate pedagogy that suits this new conception.

The message for the computer industry is that it ought to take what it knows about minis and superminis and apply that knowledge to the educational micro market. That market today is filled with me-too machines that are not much better than any other computer. The software written for such machines is equally limited and, indeed, narrow-minded. The education market will also change as the prices of microcomputers drop to the point that an increasing number of students bring them to campus. Right now, a student can buy an Apple Macintosh for about $2500 from a dealer or as little as $1000 at a school in the Apple sales consortium. Despite its very real limitations, the Macintosh has more raw computing power than most of the micros in use at colleges and universities.

The decisions schools must make about purchasing computing equipment are changing with the market. A heavy investment in today's technology cuts into the amount they can spend on tomorrow's. The school with 30 Apple IIe's in the microlab will look silly when there are 100 Macintoshes in the dorms.

The current infatuation with microcomputers in writing classes may fade quickly once English instructors become more acquainted with them. Yet, the microcomputer is a marvelous writing tool. Most writing instructors who take the time to learn how to use word processing become converts, even apostles. Nonetheless, as more and more microcomputers become available for classroom use, their limitations will become increasingly apparent. The defi-

ciencies of much current software, including many of the programs discussed in this article, will force us to ask what we as writing teachers really want computers to do. To cite one example, one of the pre-writing exercises in the *WANDAH* programs asks students to type on a blackened screen. The students cannot see what they have composed. This is an electronic version of what writing teachers call "freewriting," which is really an exercise in overcoming writer's block and in generating ideas. My question is whether freewriting on the computer is a significant improvement over freewriting on paper. If it is not any better, why do it on the computer?

Most of us who have worked with computers in writing instruction for some time are beginning to ask this question. Using computers to teach writing entails so many problems--and so many *new kinds* of problems--that we ought to examine carefully just what it is we want the computer to do for our students. Also, we must determine how the computer fits into our approach to teaching. Few people are thinking seriously about the changes we must make in our teaching strategies, even our course plans, if we are to use computers effectively. For instance, it takes time before students are truly comfortable writing on the computer. Does this mean we have to modify our course plans to take this into account?

In addition to these questions, many writing instructors are asking whether the instructional programs available today are, to put it bluntly, merely electronic versions of our worst selves. After writing my own programs and talking to others involved in projects like mine, I have concluded that English teachers are quite literally creating Frankenstein's monster. Frankenstein's monster, you recall, was a man who was not a man. He was a caricature of a man, patched together from dead parts. In the movie he was presented fairly sympathetically, but because he was so powerful he could not help killing a villager or two, usually by accident. Most of the computer programs I have seen are like Frankenstein's monster. They stress error, error, and error at a time when writing instructors are moving beyond error-based pedagogy.

There are reasons for this predicament. First, the microcomputers most such programs run on are severely limited. They are good at string matching and number crunching, and so programs like *Grammatik* look for deadwood phrases and report on sentence length, word frequency, and similar information. As I said before, unless the user knows how to interpret this information, it is useless to him. Second, writing instructors who ask programmers to help them design software usually know next to nothing about computers. More importantly, they quite literally do not know how to describe their aims to programmers. This problem cannot be explained by saying that programmers know more than writing instructors or that programmers and writing instructors are mired in mutually-incomprehensible jargon. The explanation, I think, is more subtle. Writing instructors find it hard to describe what they are looking for in terms a project programmer can understand.

What, after all, makes a piece of writing good? Writing instructors can describe the qualities of good writing in general terms, such as the harmony of form and content, or can leap into metaphor ("A grace beyond the reach of art," to quote Alexander Pope). But such descriptions fail to say much to a programmer who is used to parsing a task into a sequence of steps that must be performed to accomplish a carefully-defined goal. Writing is not that tractable. One source of the problem, then, is that writing instructors cannot clarify the qualities they are looking for in student writing in terms a programmer is likely to feel comfortable with.

Despite this problem, we usually can agree that, at the very least, good writing is usually error-free writing. Consequently, we can say we want papers without misspellings, without subject-verb agreement problems, etcetera. We know what we don't want even if we cannot define what we *do* want. When we begin to define our goals in terms of correctness, we start talking a language programmers can understand because of the relative simplicity of the problem. Even if we cannot write programs which are sensitive enough to the nuances of English to correct with infallible accuracy every slip in grammar and usage in an essay, we can invent automatic proofreaders which flag possible slips and ask the user to correct them. And so we have developed such programs, and we are developing more of them, although they don't provide complete or even accurate information about grammar and usage. As for style-analysis programs, they codify a narrow conception of style without regard for purpose, audience, or occasion. It is no surprise to the writing instructor that *Writer's Workbench* was developed to aid technical writers whose prescribed style lends itself to this kind of narrow analysis.

The current generation of programs, therefore, joins a limited technology and an obscure purpose to create tools which suit an outmoded pedagogy. This indictment sounds severe, but it is essentially accurate. It is important, though, to look beyond the limitations of today's educational software to imagine the kind of place computer-assisted instruction will earn for itself in tomorrow's classroom. As the rest of this article shall show, computers will have an increasingly important place in the writing class in at least four ways.

First, the next few years will see a boom in CAI of all sorts, but especially in writing instruction. The reason is that CAI will be increasingly cost-efficient compared to the cost of maintaining staff to teach basic skills to college students. Over the last decade the verbal skills of college students nationwide have dropped dramatically. In recent years as many as one-third of UC-Berkeley's freshman class have fared so poorly on an entrance test that they are required to complete a remedial English course. During the same years the teaching of composition established itself in many college programs as a respectable academic discipline in its own right, with the result that staffing lower-level writing courses became expensive. With colleges and universities nationwide forced to cut costs, we can expect the market for educational software to flourish.

Many schools have already established computer-assisted learning centers to supplement traditional modes of instruction. As a new generation of students comes to college already familiar with computer as a teaching tool, computers may replace or surely supplement human instructors in many courses. In writing instruction, there is no reason why a computer cannot take over the rote tasks of drill and testing in grammar and mechanics.

A more creative use of this capability of the computer is to place such information on line so writers can call it up when they need it, perhaps while editing a text file. The HELP utility on many systems can be used to create a HELP library for writing instruction. [See the November 1983 issue of the *Newsletter*.] This is one possibility I intend to explore at Ohio Wesleyan.

The second major use is more radical. It requires us to reconceptualize the computer as a teaching tool and depends upon taking advantage of the computer's communications capability. A full-scale minicomputer system is, among other things, a communications device. In my classes I use it to send and receive text files. For the writing instructor, this capability holds considerable promise. Students can examine and comment on each other's papers electronically. They can communicate with the instructor by means of the computer, as many of my students do at present. More importantly, this communication is carried out in writing (or rather in typing), which means that it is more deliberate than verbal communication. It has often happened that a student who began to send me an electronic mail message seeking my help solved his own problem before he finished typing it in. The computer makes it possible to create an extended audience for novice writers, something which they rarely have but usually benefit from. This approach to using computers overcomes one limitation of current uses: it will substitute fallible but human responses for not-quite-infallible but mechanical responses.

It should be instantly clear that microcomputer systems lack this capability. They are not usually networked together. But the utilities on many systems make it easy to use the computer as a communications device for instructional purposes. This is an area the computer industry should investigate vigorously.

Third, the computer will prove its value as a research tool for writing instruction. The keystroke-analysis programs described above explore only one of several possibilities. A computer can help us perform sophisticated protocol studies of writers to learn, in general, what goes on when people write. It can also help us understand more about the process involved in more specialized kinds of writing (e.g., technical writing). Computers with a mass storage device can retain text files indefinitely and analyze them at will. The protocol studies reported on at the 1984 Minnesota Educational Computer conference analyzed only a handful of writers: four in one study, six in another. For conclusions to be meaningful, and hence useful, the samples must be much larger. Only a computer which can store great quantities of data and manipulate that information easily is likely to produce the results these researchers are seeking. Here is another marketing possibility for the computer industry.

Fourth, on the far horizon are intelligent systems, such as the tutoring system under development at Carnegie Mellon. Admittedly, their success depends upon the likelihood of rapid advances in artificial intelligence experiments. Natural language parsers are becoming more sophisticated every day, but so far they work best with controlled texts. How they would respond to the chaos of a freshman essay is anyone's guess. Also, even if AI experiments prove successful, there is the further question of whether practical applications, such as in education, represent the best use of the enormous amount of computing resources tied up in these projects. The problem here is the reverse of the one posed by extensive use of CAI: intelligent systems are currently more expensive to develop and operate than the cost of hiring a human instructor.

Once again, however, the problem may disappear if we reconceptualize the computer. The intelligent tutoring system is, at bottom, just an electronic English teacher. The question I pose once more is why we keep inventing these Frankenstein monsters when we could be applying current knowledge to practical problems. For example, Joseph Weizenbaums' famous *ELIZA* program, now nearly twenty years old, could be converted to a "smart" inven-

tion program which initiates a dialogue with students on their topics. Again, computerized sentence-combining exercises, in which the syntactic parameters and semantic constraints can be fairly precisely specified, could be combined with AI programs in various ways. The program could detect, for instance, when students were relying too heavily on one combining strategy and then call up exercises which elicit other kinds of combinations in order to enlarge their repertoire of sentence styles. Current AI can be applied in useful ways to writing instruction.

A Look at Program Types

1. ERROR CHECKING PROGRAMS: Many faculty in writing programs use commercially available programs that search text files for mistakes of various kinds, including spelling, deadwood phrases, errors in usage, etc. The most important of these software toolkits currently available is *Writer's Workbench*. The *Writer's Workbench* project at St. Olaf's College in Northfield, MN, continues to attract attention, mostly because *Writer's Workbench* is fairly well-known to composition instructors from Kate Kiefer's widely-reported project at Colorado State. St. Olaf's has not been running the program long enough to report any hard results, but they express satisfaction with the program. Students are using it to review their essays before turning them in.

On microcomputers, *Grammatik* is the clear favorite. Ted Thiesmyer of Hobart and William Smith Colleges has adapted *Grammatik* for instructional purposes. Thiesmyer had been dissatisfied, however, with the usefulness of the information the program provides. In addition to checking a text file for various errors in usage (deadwood phrases, "to be" verbs, etc.), *Grammatik* reports on stylistic features such as average sentence length, frequency of word use, and so forth. Thiesmyer can disassemble the program and put it back together with modifications that make this information more meaningful to the student user and more appropriate to the aims of a writing class (rather than the needs of a professional writer). Along the same lines, Tommy Barker (Texas Tech) has developed a program that takes the numbers *Grammatik* has compiled and puts them into a meaningful context for student users. The program looks at the count of "to be" verbs, for example, and warns writers if they have exceeded a number set by the instructor (or praises them if they falls below it).

Despite their differences, such programs exhibit similar problems as instructional aids. First, even the researchers complain that the computer's report on errors or stylistic information is hard to interpret. The problem for naive writers is that information they cannot understand does them no good at all. For instance, *Grammatik*'s report on sentence length does not mean much to the writer who has no useable concept of sentence variety (which is in part a function of sentence length). The programs fail to provide any context for this information. Second, these programs are menu-driven. The user selects one or another program to run. This presupposes that the user has a fairly sophisticated understanding of grammar and usage. After all, a student who does not know what the passive voice is will not learn much from a program that counts "to be" verbs. Yet most beginning writers know very little about grammar and usage. They benefit very little from such programs.

Another problem is that these programs are often inaccurate in several ways. An error checker is only as shrewd as the programmer who wrote the code. Many commercial error checking programs look for "errors" that an English instructor might regard as acceptable usage. Conversely, owing to their limitation, these programs overlook many howlers. Also, these programs are spelling sensitive. They know enough to flag "due to the fact that" as a deadwood phrase, but will skip it if the student types "dew to the fact that." Again, many error checkers are likely to confuse a naive user. For example, a program developed at Denison University in Granville, OH, examines a text file for deadwood phrases, including relative pronouns (who, whom, which, that, whose, etc.). While it is true that many careless writers use too many relative pronouns and that the tendency in modern prose is to eliminate them, beginning writers often omit them where they are needed for clarity. Thus far no one has written a computer program that can tell a student where to add a missing word. Also, this program cannot distinguish between *that* as a relative pronoun ("This is the house that Jack built") and *that* as a demonstrative pronoun ("That is Jack's house"). The student who sees the computer flagging his ***that***'s is not likely to know the distinction between the two. The result may be mystification rather than enlightenment.

Third, and perhaps most important, these programs reinforce the student writer's naive and counterproductive view of the writing process as the gradual elimination of error. Most students regard their writing teachers as living, breathing red pens whose main job is to catch them out in mistakes of various kinds. There is some truth here, but few responsible college writing instructors today would affirm that their most important job is to enforce the standards of accepted usage. Most of us would say our first task is to get our students to write confidently and effectively, which we try to accomplish by familiarizing them with the process of discovering ideas, organizing them, putting them on paper, revising what we have written, and then editing our work. This process approach, as it is

commonly called, has dominated mainline research into the teaching of writing for nearly two decades. Even process-oriented instructors insist upon correct grammar and appropriate usage in student papers, but they recognize that correctness cannot be an end in itself.

The problem with programs like *Grammatik* and *Writer's Workbench* is that they foreground errors in usage instead of subordinating this information to other aspects of the process. Thus, the programs and the instructors who use them may be at odds with each other. They may stress organization or development or unity while the machine reports authoritatively on sentence length or sexist language or whatever. Many students will prefer the machine's clear and unambiguous answers to the often-fuzzy descriptions provided by writing instructors ("Your sentences don't flow"). But few novice writers are capable of learning much from the information these programs provide or even of making any meaningful changes on the basis of that information.

(An exception must be made for spelling checkers. Most instructors who have used these find them useful. Some report that dyslectic students overcome their writing anxiety by checking their text with a spelling program. Once again, this seeming exception confirms that students are handicapped by a naive conception of the writing process. Learning-disabled students find themselves blocked as writers because they know they cannot spell words correctly. The usual solution is to avoid writing altogether. Once they become accustomed to spelling checkers, they often become unblocked. An article in *The Whole Earth Software Review* attests to this.)

An ambitious project is a plan by Marist College to use IBM's *EPISTLE* program for writing instruction. If reports and rumors are anywhere near accurate, *EPISTLE* is extremely sophisticated. One person told me it can spot errors in subject-verb agreement, although it took 26 pages of programming code to define just *one* of the rules!

2. COMPREHENSIVE CAI/WORD PROCESSING PROGRAMS: Many programs (including my own *WRITER* program) combine error-checkers with computerized text editing, but none of them combines a word processing capability with a comprehensive battery of checking programs so that beginning writers can access information about their writing while they are still composing on the computer.

The *WANDAH* program from UCLA combines a text-editing program (appropriately simplified for instructional purposes) with programs that report on stylistic features, possible errors, etc. Its most innovative feature, however, is a battery of pre-writing exercises to help student writers generate material to write about. In this way *WANDAH* declares its allegiance to the process approach discussed above. Using *WANDAH*, a student can brainstorm for ideas on the computer (or try a half-dozen other pre-writing strategies), compose an essay, check it, and print it. As a UCLA product, *WANDAH* comes out of the same university as the *HOMER* program developed by Michael Cohen on the basis of Richard Lanham's "paramedic method" of sentence editing in his book *Revising Prose*. The implementation of *WANDAH* I have seen in action does not incorporate *HOMER*, but I am sure future implementations will (unless rights to *HOMER* were signed over to Scribner's, which is marketing a microcomputer version for the Apple IIe running the UCSD-p system).

WANDAH is designed to run on the IBM PC running under UCSD-p. It is extremely user-friendly. If students have to change disks, *WANDAH* tells them when and how. Most faculty who observed it are impressed because, unlike many other programs, it is pitched to the computer ignoramus and anticipates most of the problems a novice computer user might have. More importantly, *WANDAH* appears to have overcome at least one objection to error checking programs. Because it includes computer-based prewriting exercises, *WANDAH* would seem to work well in a process-oriented writing class. The UCLA team, including the project director, Morton Friedman of the Psychology Department, are enthusiastic. The program has been extensively tested, but it is not clear how UCLA plans to use *WANDAH* in its writing program.

3. KEYSTROKE ANALYSIS PROGRAMS: Many researchers have examined the differences between writing on computers and off computers. Yet it is generally agreed that we need to learn more about the way people write using word processors as this information will probably force us to modify the way we teach writing. Many researchers express dissatisfaction with protocol studies of writers using videotape or interviews or simultaneous description (in which writers write and describe what they are doing at the same time). The hope is that the computer can store all the relevant information during a writing session without interfering with the writers themselves.

Sandra Bolkema, a Ph.D. candidate at the University of Michigan, is completing a study of the use of word processors by four professional writers. She has been able to record all the changes made during their writing sessions on the word processor, and she is currently analyzing these to determine the kind and frequency of editing changes.

She was not free to present her conclusions at the present time, but her findings may be useful to software developers and computer companies.

Mark Hasselkorn of Louisiana State University described a program that preserves a sequential, timed record of a writing session. The users type text into the computer and are free to modify it any way they wish. But the computer is monitoring all of these changes. In effect, the program plays back the writing session. Hasselkorn's project is just getting underway now. The program he demonstrated was still being debugged. It is designed to run on the IBM PC. People I spoke to about it were impressed by the idea but dubious about its usefulnes. Hasselkorn treated us to a page of statistics concerning frequency of word changes, etc. But it was not at all clear how this information would help clarify our understanding of what goes on when people sit down to write. I for one felt that Hasselkorn is doomed to be swamped in data that may not tell him much at all.

This is not to say that keystroke analysis is useless. One potentially important use is in designing future word processing programs. Teachers often complain that most commercially-available word processing programs seem to be written for secretaries who transcribe text rather than for writers who create it. In other words, the word processor is conceived of as a souped up typewriter. The keystroke analysts at least recognize that a word processor alters the way we create text, the process we use. The information they uncover about, say, the frequency with which writers make particular kinds of changes could be of use to software designers. Also, their research could uncover a variety of different but definable personal styles in using word processing, with implications for software and even hardware design.

Conclusion

1. Most computer projects now underway are limited in scope and conception. They usually concentrate on error-checking and correction (e.g., spelling checkers, usage checkers, rudimentary stylistic analysis). Error correction is one aim of writing instruction, but it is not regarded as the principal aim by most leading researchers. There is, therefore, a contradiction between the capabilities of most available programs and contemporary pedagogical theory.

2. Most colleges using computers in writing classes are using microcomputers for reasons which are largely irrelevant to their merits as instructional tools. Few instructors are aware of the capabilities of a minicomputer system.

3. Most of the major projects currently under development at colleges and universities are designed for the IBM PC. Most of the proejcts underway by computer companies and by larger universities are being developed on minis, mainframes, or special-purpose micros (e.g. Lisp machines like the Xerox Dandelion).

4. The overall trend opens up new marketing opportunities for computer companies. Developing this market, though, will require skill in learning how to talk to educators. Also, educators and computer companies will gain most once they learn how to reconceptualize the computer as a teaching tool.

[ED. NOTE: Dr. Richard Elias is an Assistant Professor of English at Ohio Wesleyan University and the Director of Wesleyan's *WRITER* Project (word processing/CAI on a DEC VAX 11/750)].

Bibliography Update

Addams, Shay. "Interactive Fiction: Emphasis on Character Interaction and Story Line Makes These Text Adventures the Computer Games for the Literate." *Popular Computing*. 4:5 (March 1985), pp. 96-99, 180-182.

Anderson, C.W. and G.E. McMaster. "Objective Analysis of Emotional Tone in Stories and Poems." *ALLC Journal*. 3:2 (Autumn 1982), pp. 45-51. [Association for Library and Linguistic Computing]

Bolz, H. and G. Willee. "WISILB--A Computer Program Which Counts Syllables in English Words." *Sprache and Datenverarbeitung*. 6:1-2 (1982), pp. 43-47.

Candy, L. "User Interface Construction Software and the Computer-Aided Acquisition of Basic English Skills." *Interfaces in Computing*. 2:1 (February 1984), pp. 69-80.

Cohen, Gerald and Donald C. Cunningham. "Writing for the World of Computers." *Creating Technical Manuals: A Step-by-Step Approach to Writing User-Friendly Instructions.* New York: McGraw-Hill, 1984, pp. 119-153.

Dean, S.L. "In-House Tech Writers Guide Novice Users." *Computerworld.* 28:51 (December 17, 1984), pp. 61, 71.

Derr, R.L. "Linguistic Meaning and Language Comprehension." *Information Processing and Management.* 19:6 (1983), pp. 369-380.

Eiser, Leslie. "Can Kids Outgrow Word Processing Programs?" *Classroom Computer Learning.* 5:6 (February 1985), pp. 52-55.

Frase, Lawrence T. "Knowledge, Information, and Action: Requirements for Automated Writing Instruction." *Journal of Computer-Based Instruction.* 11:2 (Spring 1984), pp. 55-59.

Jarchow, E. McNally. "Teaching Literature with the Help of Microcomputers." *The Computing Teacher.* 11:4 November 1983) pp. 35-37.

Joyce, James. "Unix Aids for English Composition Courses." *Computing in the Humanities.* ed. Richard W. Bailey. North Holland Publishing: ICCH/5, 1982, pp. 33-38.

Kern, Alfred. "GOTO Poetry." *Perspectives in Computing.* 3:3 (October 1983), pp. 44-52.

Kuchinskas, G. "Interactive Language Experiences Between Students and Computers." *AEDS Twenty-First Annual Convention Proceedings: Frontiers in Educational Computing.* Portland, OR: May 9-13, 1983, pp. 181-184. [Washington, DC: Association for Educational Data Systems]

"Local-Area Network Frees Writers from Manual Labor." *Computerworld.* 19:2 (January 14, 1985), p. 36.

Marcus, Stephen. "Computers and the Poetic Muse." *Sixth International Conference on Computers and the Humanities.* Sarah K. Burton and Douglas D. Short, eds. New York: Computer Science Press, 1983, pp. 406-408.

Michalski, Ryszard S. et al., eds. *Machine Learning: An Artificial Intelligence Approach.* Palo Alto, CA: Tioga Publishing Co., 1983.

Mitchell, Catherine C. "Computers Are Only Machines." *The Chronicle of Higher Education.* 39:18 (January 16, 1985), p. 40.

Montgomery, J. "Cloze Procedure: A Computer Application." *The Computing Teacher.* 11:9 (May 1984), pp. 16-18, 20.

Moorman, Charles. "Computing Housman's Fleas: A Statistical Analysis of Manly's Landmark Manuscripts in the General Prologue to *The Canterbury Tales.*" *Sixth International Conference on Computers and the Humanities.* Sarah K. Burton and Douglas D. Short, eds. New York: Computer Science Press, 1983, pp. 431-446.

Nisenholtz, M. and E.M. Morphos. "Information Technology and New Forms of Participation in the Narrative." *Information Processing 83: Proceedings of the IFIP 9th World Computer Congress.* Paris, France: September 19-23, 1983, pp. 727-731.

O'Brien, P. "Using Microcomputers in the Writing Class." *The Computing Teacher.* 11:9 (May 1984), pp. 20-21.

Paikeday, Thomas M. "The Joy of Lex." *Creative Computing.* 9:11 (November 1983), pp. 240-245.

Palmer, A. et al. "Changing Teacher and Student Attitudes Through Word Processing." *The Computing Teacher.* 11:9 (May 1984), pp. 45-47.

Pearlman, Dara. "Throw Out Your Index Cards: Here's How To Use Your Editing System To Take Notes, Keep Files, and Build a Database for Research or Writing Projects. It's a Lot More Efficient Than Shuffling Index Cards." *PC: The Independent Guide to IBM Personal Computers.* 4:4 (February 19, 1985), pp. 331-332.

Piper, K. "Word Processing As a Tool for Structured Writing Instruction with Elementary Students." *AEDS Monitor.* 23:1-2 (January-February 1984), pp. 22-24. [Association for Educational Data Systems]

Pollitt, A.H. "Warming to the Wonder of the Word Processor." *The Computing Teacher.* 11:9 (May 1984), pp. 48-49.

Porter, Martin. "Oxford Goes On-Line: As the Editors of the Huge and Authoritative Oxford English Dictionary Put All 17 Volumes On-Line, IBM Is Providing Funds and PCs for the Project." *PC: The Independent Guide to IBM Personal Computers.* 4:4 (February 19, 1985), pp. 233-238.

Redman, Timothy, Deborah H. Holdstein, and L. DeCelles. "Evaluating Computer-Assisted Instruction for Writing: A First Glance." *Collegiate Microcomputer.* 2:2 (May 1984), pp. 101-106.

Schantz, L.M. "The Computer As Tutor, Tool and Tutee in Composition." *The Computing Teacher.* 11:3 (October 1983), pp. 60-62.

Schwartz, Helen J. "Pre-Writing with SEEN, a Program for Hypothesis Testing." *AEDS Monitor.* 23:1-2 (January-February 1984), pp. 20-21. [Association for Educational Data Systems]

Shillingsburg, Miriam J. "Relying on the Weird: Dangers in Editing by Computer." *Sixth International Conference on Computers and the Humanities.* Sarah K. Burton and Douglas D. Short, eds. New York: Computer Science Press, 1983, pp. 654-658.

Shneiderman, B. et al. "An Empirical Comparison of Two PLATO Text Editors." *Journal of Computer-Based Instruction.* 10:1-2 (Summer 1983), pp. 43-50.

Skellings, E. "Rhyming To Reason." *Perspectives in Computing.* 3:4 (December 1983), pp. 12-19.

Stewart, George. "The Text Scanner: Program Your Computer To Test the Readability of Any Writing Sample." *Popular Computing.* 3:9 (July 1984), pp. 199-203.

Turner, Kathleen and Matthew Marino. "Beyond Word Crunching." *Sixth International Conference on Computers and the Humanities.* Sarah K. Burton and Douglas D. Short, eds. New York: Computer Science Press, 1983, pp. 717-724.

Ule, Louis. "Recent Progress in Computer Methods of Authorship Determination." *ALLC Bulletin.* 10:3 (1982), pp. 73-89. [Association for Literary and Linguistic Computing]

Vincent, G. "Computhink: Software for Learning English." *Micro Systemes.* 36 (November 1983), pp. 190-192. [in French]

Wheeler, P. "Micros Made Easy: Microwriting." *Educational Computing.* 5:2 (February 1984), pp. 26-27.

Wresch, William. "Essay Writer: A Program To Help Students Through the Writing Process." *Collegiate Microcomputer.* 1:3 (August, 1983), pp. 281-287.

--B.A.M.

Conference on College Composition and Communication

The CCC convention in Minneapolis will begin March 20, 1985, with a preconvention workshop entitled *Word Processing in the Classroom*--chaired by Lillian Bridwell and Donald Ross--and end March 23rd with a postconvention workshop called *Computer Labs for Writing Classes*, chaired by Helen J. Schwartz. USAF Major Hugh Burns will be a featured speaker for the main convention, drawing upon his experience with computer-assisted invention software and his research on artificial intelligence in the writing process. At least fifteen other papers and meetings during the main convention will focus on applications of word processing to the writing process:

Thursday, March 21: ■ Revising and Editing with Computers ■ From Cro-Magnon Man to Computers: The Past, Present, and Future of Systems of Notation ■ Computer Software; Critiques and Caveats ■Beyond Word Processing: Other Uses for Computers ■ Collaborative Writing and the Computer ■The Fifth C: Computers

Friday, March 22: ■ Dynamics of Computer-Assisted Writing ■ New Approaches to Text Analysis ■ Intelligent Computer-Assisted Instruction in Writing: Teaching Procedural Skill ■Inventing, Evaluating, and Analyzing with Computers ■ The Other C: Composition, Communication, and Computers in the Writing Center ■ Computers and the Writing Process

Saturday, March 23: ■Research on the Use of Computers in Composition ■ A Comprehensive Computer-Aided Program in Writing ■ RECOMP: Computer Analysis of the Composing Process

THE SCHOLAR'S SOFTWARE LIBRARY

Bryan Pfaffenberger

Program:	*Microsoft Word*
Available from:	Microsoft Corporation 10700 Northup Way Bellevue, WA 98004
Price:	$520 (with Microsoft Mouse); $375 (without Microsoft Mouse)
Requires:	IBM PC or PC-Compatible with 192K of RAM (version 1.15) or 256K of RAM (version 2.0) and one double-sided, double-density drive; all versions require DOS 2.0 or higher.
Recommended:	256K of RAM; two double-sided, double-density drives; color graphics display adapter and composite monochrome monitor (or Hercules Graphics Card and IBM Monochrome Monitor)
Applications:	Full featured word-processing program

Microsoft Word is Microsoft's big, full-featured answer to *WordStar*, and--like *WordStar*--people tend to be ambivalent about it.

On the one hand, *Word* includes just about every snazzy feature that you'd ever want, including on-screen display of italics, boldface, etc., and the full IBM PC character set; an extensive and intelligent implementation of the optional mouse for program control and editing; multiple windows and the ability to work with more than one document at once; glossary buffers that can handle huge chunks of text; and an exceptionally well implemented set of printer drivers for more than two dozen printers (including Hewlett-Packard's LaserJet).

On the other hand, its screen display is busy (you have to write inside a glowing rectangle), it doesn't use standard DOS files, and--worst of all--it's a monster of a program to learn. The program comes in a huge box, and your first instinct on opening it is to apply for sabbatical leave so you'll have enough time to learn how to install the program disk. For undergraduates, the program (again, like *WordStar*) could well prove so daunting that they'd be inclined to abandon word processing altogether. That said, there are two reasons that *Microsoft Word* deserves the consideration of graduate students, academics generally, and professional writers: it can do footnotes and endnotes the way professional writers have to do them. And--even better--it lets you define your own, customized keyboard commands to give you precise control over text formatting.

Footnotes and Endnotes

Word's footnote feature is an academic's dream come true. To insert a footnote, you use one of the commands arrayed at the bottom of the screen, and the program gives you an opportunity to provide a reference mark (such as an asterisk). If you make no response, the program assumes you want the notes numbered automatically. Next, you jump to the footnote window, a special area at the end of the file for footnotes. The reference mark you've entered (or, if you've made no response, the note's number) is displayed, and you may then type the note. Another command takes you back to the text, right where you left off.

If you wish, you may open the footnote window so that it's visible at the bottom of the screen, and there's much to be said for doing so. The footnote window is smart: it automatically shows you the note (if any) that's cited in the text you're displaying in the main window. This feature makes it a cinch to edit your manuscript accurately. The footnote relevant to the text you're editing is always right there, and it's much less likely that you'll forget to make the necessary changes in your footnotes after editing the text.

Word lets you format footnotes just the way you want them formatted: you can double-space them, indent them, and include italics and boldface in them. And with one simple command, you can position them at the bottom of the page or at the end of the manuscript.

1		Normal division a	Normal Document Setup
		Page break. Page length 11''; width 8.5''. Folios Arabic. Top margin 1.1''; bottom 1.1''; left 1''; right 1''. Top running head at 1''. Bottom running head at 1''. Footnotes on same page.	
2	P	Normal paragraph a	Normal Paragraph Setup
		TrendPS (roman b) 12. Flush left. Tabs at: 0.4'' (left flush), 0.9'' (left, flush), 1.4'' (left flush), 1.9'' (left flush), 2.4'' (left flush), 2.9'' (left flush).	
3	C	Heading a	Centered Text
		TrendPS (roman b) 12. Centered, space after 1 li.	
4	J	Other paragraph a	Justified Text
		TrendPS (roman b) 12. Justified.	
5	R	Other paragraph b	Flush-Right Text
		TrendPS (roman b) 12. Flush right	
6	T	**Other paragraph c**	**Bibliographic References**
		TrendPS (roman b) 12/24. Flush left, Left indent 0.5'' (first line indent -0.5'').	
7	2	Other paragraph d	Double-spaced Text
		TrendPS (roman b) 12/24. Flush left.	
8		Other character a	Normal Font
		TrendPS (roman b) 12.	
9	B	Emphasis a	Boldface Font
		TrendPS (roman b) 12 Bold.	
10	K	Emphasis b	Small Capital Letters Font
		TrendPS (roman b) 12 Small caps.	
11	S	Emphasis c	Strike-Through Font
		TrendPS (roman b) 12 Struck through.	
12	U	Emphasis d	Underlined Font
		TrendPS (roman b) 12 Underlined.	
13	D	Other character b	Double-Underlined Font
		TrendPS (roman b) 12 Double underlined.	
14	=	Superscript 1	Superscript Font
		TrendPS (roman b) 12 Superscript.	
15	-	Subscript 1	Subscript Font
		TrendPS (roman b) 12 Subscript.	
16	F	Footnote reference a	Footnote Numbers Superscript
		TrendPS (roman b) 12 Superscript.	
17	N	Footnote a	Footnote Reference Format
		TrendPS (roman b) 12. Flush left, space before 1 li.	

Fig. 1: *Microsoft Word* Sample Stylesheet

Best of all, *Word*'s footnote feature *works*. You'll never get an error message that the program couldn't handle the note if it occurred too near the top or the bottom of the page (all too common in certain programs with "footnoting capabilities").

Defining Your Own Keyboard Commands

Full-featured word processing programs almost always use keyboard commands, or commands entered by holding down the [CTRL] or [ALT] key and pressing another key, for formatting text. With *WordStar*, for example, you use the [CTRL] OS (LOS) command to set line spacing. Before *Microsoft Word*, however, you were stuck with whatever formatting commands the programmers felt you needed--and, more than likely, they failed to anticipate your needs.

Like other full-featured programs, *Word* comes with a built-in set of keyboard commands for formatting text and, if you wish, you can use them just like you'd use any other program's commands. But what makes *Word* so appealing is its stylesheet feature, which permits you to define your own keyboard commands. Imagine, for example, entering the command [ALT] R (for, say, "Reference List") and then typing your reference list formatted precisely according to a journal's style guidelines (for example, double-spaced with a 0.5" hanging indent; **SEE #6 IN FIG. 1).**

What's particularly exciting about *Word*'s stylesheet feature is that you can piggy-back several formatting characteristics on the same keyboard command. The command [ALT] TA (for "Table"), for example, could be defined so that it makes available, whenever entered, a detailed pattern of tab settings, single line spacing, and slightly widened margins. The command [ALT] H1 (for "Heading 1"), for major headings, could be defined so that it positions the heading ten lines down the page, centers it, and boldfaces it.

Microsoft Word, in sum, has much to recommend it to academic writers. If you've decided that you need a full-featured, word-processing program (and you're willing to invest some time in learning it), *Word* deserves very serious consideration. In a future column, we'll have a look at MicroPro's new entry in the word-processing sweepstakes--*WordStar 2000*--to see how it measures up to *Word* from the academic writer's point of view.

[ED. NOTE: Bryan Pfaffenberger--an anthropologist, college teacher, and writer--is the author of *The College Student's Personal Computer Handbook* and *Macintosh for College Students* (Sybex Computer Books, 2344 Sixth St., Berkeley, CA 94710), as well as *The Electronic Scholar*, to be published in 1985 by Little, Brown and Company. He is currently on leave from his position as Associate Professor of Anthropology at Knox College.]

Manuscript Submissions Welcome

The newsletter welcomes article submissions which pertain to word-processing, text-analysis, and research applications in academic situations. Manuscripts may be submitted either as hard copy or on disk using *WordPerfect*, *WordStar*, or standard ASCII code in IBM PC-DOS, MS-DOS (5¼" diskettes), or CPT 8500 (8" disk) formats. The Editors reserve the right to edit articles, if necessary. If you want your manuscript or disk returned, please send enough postage to cover the return along with a self-addressed envelope. Address all correspondence to the Editors, *Research in Word Processing Newsletter*, South Dakota School of Mines and Technology, 500 E. St. Joseph, Rapid City, SD 57701-3995.

Research in Word Processing Newsletter. Volume 3, Number 3. Copyright © 1985 by the South Dakota School of Mines and Technology. All rights reserved. ISSN: 0748-5484. Also indexed by ***ERIC*** and ***IHSPEC.*** The *Research in Word Processing Newsletter* is published 9 times a year (September through May) by the South Dakota School of Mines and Technology, Rapid City, South Dakota 57701-3995; telephone (605) 394-2481. Postage paid at Rapid City, South Dakota. **SUBSCRIPTIONS:** $12 per year (US); $18 per year (Canada); $24 per year (foreign). Address all subscription inquiries and manuscripts to the Editors, *Research in Word Processing Newsletter*, South Dakota School of Mines and Technology, 500 E. St. Joseph, Rapid City, SD 57701-3995. Please allow four to six weeks for new-subscription processing.