Virtual Muse

Charles O. Hartman

Published by Wesleyan University Press

>>>> / 4 /

THE SCANSION MACHINE

A search of the catalog in a big library turns up quite a few cross-references between "computers" and "poetry." But virtually all of the books and articles referred to have to do with "computer stylistics." That is, they're documents in the field of literary criticism, and they represent endeavors to *study* poetry by means of computers, not experiments in *making* poetry with computers.

Computers are well suited to some tasks that are useful to certain kinds of literary criticism. In my college computer class, one project the teacher suggested was a dictionary program that would organize a list of words so that a particular one could be retrieved quickly. The methods involved—searching and sorting—are the bread and butter of computer science and have been exhaustively researched. These are computer capabilities just there for the asking. If the study of literature can use them, it's welcome.

And it can. The classic literary computer project is the concordance: a list of all the words that appear in a certain work (or all the works by a certain author or, in the grandest concordance so far, the *Thesaurus Linguae Graecae*, the whole of classical Greek literature). The list is arranged in alphabetical order for easy reference, preferably with an indication of the context for each appearance of the word. Concordances used to be prepared painfully by hand, usually by many hands. In the precomputer age, therefore, they existed only for a small number of very important books: the Bible, Shakespeare, the complete works of a handful of poets.

Nowadays creating a concordance is vastly easier. The methods for converting the text into a list of entries and arranging these in acces-

sible order are well understood. Programs for doing these things circulate more or less freely in the academic publishing community. The job can even be done on a home computer, though the first and last stages are tedious. Before the computer can break down and sort the original text, someone has to put it into machine-readable form. This remains a large, labor-intensive effort. A book long enough to be worth a concordance would take days or weeks for one person to type in, and perhaps months to check for accuracy. Automatic text scanners can mechanize the big job of input, but even good OCR (optical character recognition) programs, which can read a wide variety of typefaces and deal with defective printing, still need scrupulous proofreading. (Many advertise 99 percent accuracy, but that's a couple of dozen errors per page.) And once the concordance is made, publishing the finished book isn't a family-room job. (Actually, concordances are less and less often printed as books at all but "published" over computer networks.)

Aside from helping you find a passage you half-remember, a concordance can be informative. If you want to know what the Bible means by the word *covenant*, a good first step would be to read all 481 passages that use it. For this reason, I wrote an *interactive* concordance generator. It shows you a text (that you've put "on-line") and lets you browse, choosing particular words to "concord," which it can do very quickly. Each instance of the word or phrase appears in the middle of a line of context—a KWIC, or Key-Word In Context concordance. You can sort these lines in many ways to see what words follow or precede the one concorded. This kind of study of a long poem, for instance, can reveal unsuspected patterns. If *and* almost always follows a comma, the poet favors compound sentences more than compound phrases. If *be* ends many sentences, the poem is bound to feel "philosophical."

Furthermore, the kind of information found in a full concordance can be generalized into a kind of statistical characterization of an author's style. (Hence, the term computer stylistics.) How large is his or her vocabulary? How specialized? What's the ratio of active to passive verbs? How often does she or he turn nouns into adjectives?

What proportion of the verbs are modified by adverbs? This kind of authorial profile can sometimes help to identify the author of a doubtful work or straighten out the chronology of an author's writings.

I've never found this species of scholarship the most interesting kind of literary criticism to do, though I'm glad to benefit from its results. It does appeal to some part of me—just as sometimes, when an important problem is bubbling on the back burner, I enjoy making indexes of my music collection. Order and information have their own appeal, beyond usefulness. Computers offer endless opportunities for this kind of fiddling. Like most laborsaving devices, they create as much work as they perform. (So are fears about automation destroying jobs groundless in the long run? Well, it's in the short run that people need to eat.)

But beyond such distractions, statistical analysis may provide insights that again have to do with the boundary between the mechanical and the creative. Counting verbs, for instance, is mechanical. But discovering that Shakespeare activates his language by making new verbs out of nouns is a creative act of reading. No simple sum of mechanical acts will automatically generate the discovery, yet the counting is a path toward the discovery. Furthermore, counting (if it's done with intelligence and imagination) can be used to prove a more general thesis than a single flash of intuition. Walter Jackson Bate, in his biography of Samuel Johnson, pursues the clue of a very high percentage of active verbs in Johnson's poetry toward a comprehensive picture of Johnson's surprisingly powerful use of abstract language.

Stylistics is not my field. But one of my fields has been *prosody*: the study of poetic meter and rhythms. So one of the first large computer projects I undertook was a Scansion Machine.

Scansion means marking the way a line of metrical verse uses its underlying meter. The meter is an unchanging abstraction; but any particular line realizes and varies the pattern, and that produces the rhythmic interest of metrical poetry. Scansion helps show the relation between general meter and particular rhythm.

There are two main ways to scan verse these days. There's the traditional method, like this:

x / | x / | x / | x / | x /

The curfew tolls the knell of parting day

It sees the line as a series of *feet*, conventional rhythmic groupings of stressed and unstressed syllables. There's also a newer method, originated by Morris Halle and Samuel J. Keyser, based on generative linguistics. From their general study of stress in English words and phrases, Halle and Keyser derive formulas that determine whether a line is metrical or unmetrical.

For the Scansion Machine, I chose the older method for several reasons. First, it requires a bit less lexical information—less elaborate cataloging of how individual words are formed and pronounced. Therefore, it's less "data-intensive" and more "computation-intensive." The important task in traditional scansion is figuring out how the line is made up of its constituent units, or feet (iambs, trochees, etc.), rather than compiling a lot of data about particular words. But both methods require some lexical data, and this difference between them is small.

Second, I wasn't convinced that the newer method has as much to do with how people actually read poetry. Meter is a body of conventions, a set of mutual understandings between the poet and the reader, more than a matter of unconscious linguistic knowledge (like syntax). In this sense, the very fact that traditional scansion is traditional gives it an advantage of accuracy.

Third, while the new method did lend itself in an obvious way to computerization—after all, generative linguistics emulates mathematics as closely as possible—the traditional method could be thought of more as an art in itself. Experts in the field often disagree about particular scansions, and their opposing arguments about a passage involve aspects of the language far beyond the strictly lexical. In making a traditional scansion, you can't entirely ignore the meaning of the passage. This makes it more plausible to argue that the scansion, in turn, will tell you something about the meaning. That makes scansion a tool of literary criticism of an interesting kind. And

it also focuses attention, again, on the boundaries of what is computable.

Schoolchildren used to learn how to scan verse. Today, graduate students in English often have to be taught it. The program I wrote will scan iambic pentameter about as well as a good student after a semester's work.

When I set out to teach the computer to scan, naturally I drew on my experience of teaching people to do it. I had worked out a more or less foolproof order of steps to follow in scanning a line of iambic pentameter. Whether I myself follow these steps in order when I scan poems remains an open question. But here's how a student or a program can set out to do it:

1. Look up in the dictionary all the words with more than one syllable, to find out where the main stress in each one falls, and mark it. (/). For the computer, this means asking the user about syllables and stress position in each unknown word and storing the answers in an internal dictionary.
2. Place stresses (/) on the obviously important one-syllable words—generally, all nouns, verbs, adjectives, adverbs, and interjections, but not conjunctions or pronouns or prepositions. For this purpose, the computer asks the user about the unknown word's part of speech and again stores this in its dictionary.
3. Mark all the rest of the syllables as unstressed, or "slack" (x). To do this, you again have to know how many syllables the word has. This is difficult to deduce in English (spelling is quirky and linguistic history is tangled: a word like *aged* may be one syllable or two depending on whether it describes a man or a cheese), but the information is now stored in the computer's dictionary.
4. Divide this string of preliminary marks into feet (|). This is the hard part, where logical complexities and special knowledge come in. Both the student and the computer expend most of their energy and make most of their mistakes in this stage.
5. Write out the finished scansion above the line itself. This is a

trivial problem for the student. But the computer, to line the marks up with the words, must store not only the number of syllables but the positions of the boundaries between them, must deal with blank spaces, punctuation between words, and proportionally spaced fonts. Here's where the human, always a whiz at visual pattern recognition, beats the pants off the computer.

The difficulties of step 4 are the especially interesting ones. To see why, let's go back to that important distinction between *meter* (which is an abstract pattern

de-DUM de-DUM de-DUM de-DUM de-DUM

that lies unchanging behind all the individual lines) and *rhythm*, which varies from any particular line to any other. In "The curfew tolls the knell of parting day" (Thomas Gray), the similar sounds and meanings of "tolls" and "knell" clinch the line around a strong center. In "The trim of pride, the impudence of wealth" (Alexander Pope), sound and grammar group the first and third nouns and the second and fourth. This and a dozen other details distinguish the rhythms of the two lines, though they're identical in meter.

(We're about to stray into some technicalities involving lovely but inscrutable words like *anapest*. If these don't ring a bell, the reader can take them as magic spells and not worry. The general line of argument, which I hope will be clear, is what's important.)

The influence of the speech-based rhythm on the abstract pattern of meter shows up in "metrical substitutions." Trochees (/ x), anapests (x x /), spondees (/ /), or other combinations get substituted for the predominant iambs (x /). Here's a line by W. B. Yeats (from "No Second Troy") with three substitutions:

x / | x / | x x / | / / | xx /

Have taught to ignorant men most violent ways

The syncopated rhythm of the words pushes against the regular beat of the metrical pattern, forcing an accommodation that shows up in the third, fourth, and fifth feet of the scansion. When we listen to the

line in its context in the poem, we hear both rhythm and meter at once, jostling against each other in a lively way. Among other things, the line's metrical handling tells us to equate "ignorance" with "violence."

Conversely, the metrical pattern influences the rhythm the line has when spoken. This shows up in "promoted stresses"—syllables not much emphasized in speech but stressed by the meter. In scansion these are sometimes marked "(/)"; but to facilitate the final step (5) in the sequence (printing the result), I used a one-character mark: "%". Here's a line of Wordsworth's with two promoted stresses (as well as one substitution):

x / | / / | x % | x / | x %

A sight so touching in its majesty

"In" and the last syllable of "majesty" aren't strongly stressed, but the meter encourages us to hear them as more strongly stressed than the syllables that flank them.

Because of the two kinds of possible variation, a line can be *irregular* without being *unmetrical*. In English metrical verse, this kind of licensed rule-bending is a main source of rhythmic liveliness. Sometimes, as in Yeats's line about ignorant violence, the variations can very directly create meaning.

These two kinds of variations—substitutions and promotions—create many ambiguities about where to place the divisions between feet in an irregular line. Both the number of syllables and the number and position of stresses can vary. Most of these ambiguities can be resolved by means of a set of rules—not a brief set but certainly not infinite in number or in subtlety.

These rules, deduced from the practice of poets throughout history and sometimes no more than statistical generalizations, are what I was interested in making explicit by building them into the logic of a program. The procedure called FootDiv was by far the longest in the program. I'll give just one concrete example of the kind of problem that took a while to iron out.

The program checks first to see if a line is "headless"—that is, if it begins with a single stressed syllable. This isn't uncommon. If the

line is only nine syllables long and if the first three syllables are / x /, then it's a good bet, and the program proceeds on that assumption:

/ | x / | x / | x / | / /

Autumn gives us hours to think things through.

But suppose the line begins with /x/ but is ten syllables long, like a normal iambic pentameter? There are two main possibilities. It might go like this:

/ x / x / x / x x /

Winter shakes the restless fire in the grate

Or it might go:

/ x / / x / x / x /

Give me three reasons why the summer ends

or any of several other variations. The first one should be treated as headless:

/ | x / | x / | x / | x x /

Winter shakes the restless fire in the grate

(Why not "/ x | / x | / x | / x | x /"? Because even two trochaic substitutions in a row are quite unusual, and four would make the line hopelessly unstable. The "headless" reading is much more natural, meaning that it's a much simpler match between the rhythm and the metrical pattern.) But the second line—

/ x | / / | x / | x / | x /

Give me three reasons why the summer ends

—won't work if you try to make it headless. (One of the remaining feet will have to be either x / / or / x /, neither of which is a "legal" substitution. Legal? Well, rarely. The first is called a bacchius and is occasionally found as a substitution for the anapest (x x /) in verse whose normal base-foot is the anapest. The second is called a cretic or amphimacer and does show up occasionally in some irregular iambic verse. The Byzantine complexity of traditional metrical theory can itself be an idle pleasure. Even more, in some moods one can love the wonderful sound of all the *names*.)

How can the program tell these two instances apart without degenerating into an endless list of special cases? The best answer turns out to be a two-stage approach. In the initial test for headless lines,

reject all of these ten-syllable cases; treat them as normal. The second example ("Give me three reasons") can be scanned successfully that way. But at the end of the foot-division process, the first line ("Winter shakes") will be caught by a new final test: Any line that begins with more than one trochee in a row gets turned into a headless line instead. So / x | / x | / x | / x | x / will be corrected to / | x / | x / | x / | xx/.

All these shenanigans required a program about eight hundred lines long—not all that large. (The routine that handled the internal dictionary took up almost as much of it as the foot-division calculations.) It performed surprisingly well. Here's Shakespeare's Sonnet 116:

/ | x / | x x / | x x | / /
Let me not to the marriage of true minds
x / | x / | x % | / x | / /
Admit impediments; love is not love
x / | x % | x % | x / | x /
Which alters when it alteration finds,
x / | x % | x / | x % | x /
Or bends with the remover to remove.
/ / | x % | x / | x / | x /
O no, it is an ever fixed mark
x / | x / | x % | x / | x / x
That looks on tempests and is never shaken;
x% | x / | x / | x / | x x /
It is the star to every wandering bark,
x / | x / | x / | x / | x / x
Whose worth's unknown, although his height be taken.
/ / | / / | x /| x / | x /
Love's not Time's fool, though rosy lips and cheeks
x / | x / | x / | x / | x /
Within his bending sickle's compass come,
/ / | x / | x x | / / | x /
Love alters not with his brief hours and weeks,

x / | x / | / x | x x / | x /

But bears it out even to the edge of doom.

x % | x / | x % | x / | x /

If this be error and upon me proved,

x / | x / | x / | / / | x /

I never writ, nor no man ever loved.

This is not an easy poem to scan, as my students have often complained. And there's nothing in the computer's scansion that I'd mark *wrong* in a student paper, though there are lines that I would scan differently. (I read the fourth line as "Or *bends with* the remover to remove.")

Even a highly irregular poem like Yeats's "The Second Coming" falls pretty well within the program's scope. it deals calmly with lines like these:

/ x | x / | x % | x / | x x /

Turning and turning in the widening gyre . . .

and

x / | x % | x x / | x % | x /

The ceremony of innocence is drowned . . .

and even

x / | x / | x / | x x / | / /

The Second Coming! Hardly are those words out

x x | / / | x / | x / | x x / x

When a vast image out of Spiritus Mundi

/ x | x / | / x | x / | x x / x

Troubles my sight: somewhere in sands of the desert . . .

But I would quarrel with the program's handling of this line from Yeats's "No Second Troy":

x x | / / | x % | x x / | x %

That is not natural in an age like this . . .

And in "The Second Coming," the program prints an "I GIVE UP" message at the end of the line, "A shape with lion body and the head of a man."

These two different failures have a common cause. The line from

"No Second Troy" would be more accurate if the promoted stress fell on the monosyllabic "in," rather than on the end of "natural." The line from "The Second Coming" can't be scanned at all unless the end of "body" and the "and" are *elided*—treated as a single unstressed syllable—which depends on the knowledge that "body" ends with a vowel and "and" begins with one and that elision has been conventionally acceptable in various periods of literary history.

Why can't the program learn these details? Or rather, why would it have to be radically rebuilt to learn them? Adding a new idea like elided syllables isn't that hard for a human student.

One answer is that most computers can only do one thing at a time, and this forces the programmer to break a process down into discrete steps. (To a smaller degree, teaching encourages the same thing. That's how I came up with my list of five steps toward scansion.) When you add the steps before and after the actual scansion that the computer needs to have made explicit, the full procedure includes

1. reducing the words to a preliminary series of stress and slack marks with the help of the dictionary.
2. deducing from the line of marks an array of five foot-codes,
3. retranslating the foot-codes into expanded and corrected scansion marks, and
4. correlating the final scansion marks with the words again.

But this procedure means that lexical information, information about the words *as words*, is invisible in the middle stages. And the problems in those two lines involve exactly that relation between the scansion marks and the words themselves.

The computer could theoretically be made to switch gears at this point—to go back to a lexical view of the line and use that to correct the narrow logic of scansion marks. It would be a big project. But more important, I don't know how to tell the computer *when* to do this. When is lexical information relevant (aside from these two particular cases I happened to run across)? I don't know how I know that, so I don't know how to tell the computer to decide. There are

"expert system" AI programs that handle this kind of problem, a what's-relevant problem. (Medical diagnostics is an area where this approach gets applied.) These are, to say the least, bigger programs than I'd like to undertake.

Even within the foot-division logic, the order of different tests and operations makes a tricky difference. Go back to that problem of the "headless" line I mentioned earlier. The first line of Shakespeare's sonnet, "Let me not to the marriage of true minds," presents metrical problems even for human scholars. But pity the computer that had to follow these steps, as in an earlier version of my program:

1. Detect / x / at the opening and mark off a defective foot plus an iamb, no matter what the line's length.
2. Now, seeking three feet among seven remaining syllables,

   x x / x x / /

   ("to the marriage of true minds")

   take the one extra syllable as a clue to look for an anapest (x x /).
3. Pick the pattern / x x / as the best candidate for the position of the anapest: / | x x /. (This rule usually works quite well.)
4. Identify the preceding foot as another anapest—the only possibility.
5. Give up in despair on the last foot:

   / | x / | x x / | x x / | ?

   Let me not to the marriage of true minds

Failures of this kind led me to revise the sequence this way:

1. Detect / x / at the beginning, but note that the line is not a syllable short (like the simplest kind of headless line).
2. Finding no other major exceptions, divide the line into two-syllable units as usual: / x | / x | x / | x x | / /
3. Reviewing the completed scansion, notice the highly unusual double trochees at the beginning. Replace them and the succeeding iamb with a defective foot, an iamb, and an anapest: / | x / | x x / | x x | / /

Experiment shows that this system discriminates more accurately between different cases. The rules it embodies turn out to be

more general, applicable to more lines. Even so, some literary critics (Helen Vendler, notably) might argue convincingly that the more unusual scansion, the double-trochee opening, is the right one for this unusual, abrupt opening line. The computer can't join in or settle debates of that kind—though by forcing us to specify the rules explicitly, it might have a few points to contribute.

Even when it comes to problems of the other kind, flaws in the very structure of the program's algorithm (or computational procedure), I'm not saying that solutions are impossible, that we've found the point where scansion becomes noncomputable. Instead, the solutions would require a different approach to programming and another order of programming effort. That in turn tells us something about the kind of knowledge we're bringing to bear when we scan a line of verse.

The most interesting point about this program may be the way it partially refutes my original assumption. I began by saying that traditional scansion requires an awareness of the meaning of the words. As we've seen, I can occasionally trace a disagreement between my own scansion and the computer's scansion to its failure to *read* the line in any real sense. Where its logic produces this,

x / | / x | /x | x % | x /

A gaze blank and pitiless as the sun,

I might note instead how sound and meaning isolate the adjective "blank," and scan

x / | / | x /| xx % | x /

A gaze blank and pitiless as the sun.

Similarly, in Shakespeare's sonnet, it's only a feeling for what the line means that would make us read "That bends *with* the remover to remove," rather than the more regular rhythm scored by the computer's scansion. Yet in the large majority of cases, the mechanical application of foot-division rules succeeds as well as the subtlest human sensitivity to rhythmic nuance. As a system, scansion usually just isn't complicated enough to need a human being.

Once I had finished the Scansion Machine to this level of performance, I was at a loss as to what to do with it. This was the purest of

pure research. What use did it have? It wasn't likely to become a software bestseller. I could distribute it as a public-domain program, but to whom? Who needed an automated iambic pentameter scanner?

Nobody. If the program had a purpose, it was to show that it could do what it did: that traditional scansion could, within certain interesting limits, be automated. The logical next step wasn't to distribute the program but to present it in an article in some technical journal—if I could find the right sort of journal—which I never got around to doing.

But the programming work I had put into my Scansion Machine could go to another purpose as well. It might be linked to my second big programming project. Though it was too big and never finished, it's a relevant one to describe.

The second project was a text editor—"word processor" would be too grand a term—meant especially for poets. It had several features that writers of verse would prize, though nobody else had ever missed them in the many excellent word processing programs available commercially. Since verse is by definition language in *lines*, my program's special ways of manipulating text had to do with the kinds of things poets do with lines.

Part of the legacy of Modernism, especially the American version of Modernism we owe to William Carlos Williams, is an understanding of how much *lineation* has to do with what a poem means. To a reader who is listening, there's a big difference between

when I turn
my eyes away

and

when I turn my eyes
away

Writers of "free verse" often revise poems in important ways without changing a word, just by changing the way the lines are broken and distributed on the page. The lineation helps the poet control how

the reader hears the poem, and carefully controlled rhythm governs the very meaning of the words that make it up.

So poets spend a lot of time manipulating lines. Most word processing programs aren't set up for this. They're meant for prose, and they work best on prose units—words, sentences, paragraphs. To do something like

dropping part of a line down vertically takes quite a
few keystrokes. To break a line
In the middle (and perhaps
Capitalize it), which is a single
logical operation
In the mind of the poet, requires

several separate steps. Rejoining two lines is even more tedious. When one mental operation has to be broken into a series of stages, the writer's concentration gets dissipated. So my text editor made these line manipulations easier.

But it went beyond that by incorporating the main feature of the earlier Sinclair ZX81 program: randomness. This time the point wasn't to generate random poems. The point was for the writing machine—a sort of glorified typewriter—to offer the poet as much help as it could in its humble mechanical way. One kind of help that poets do sometimes need is a kind of *jolt*, something to stir the mind's waters out of lassitude and placidity.

Traditionally, the stringent requirements of meter and rhyme have played this part. Having to match a rhyme often makes the poet shift the poem's stream in an unintended, serendipitous direction. Some poets use other texts as springboards, either incorporating them as quotations or taking off from them in ways that the reader never sees.

My new version of this technique was called the Hoard. You could type in a lot of fragments (phrases, single words, half-lines) and store them in a sort of treasury. Then, when you were writing a poem, if you got stuck and wanted an external, unpredictable impulse, one keystroke would call up a fragment from the hoard at random and

insert it into your poem. Of course, you might immediately delete it (there was a one-key command for that, too). But in the meantime it might have given you another, more suitable impulse.

I called the whole program, including the Hoard and the text editor, AleaPoem. "Alea" is the Latin word for "dice"; aleatory composers leave some elements of their music or its performance up to chance processes like rolling dice.

My next thought was to incorporate the Scansion Machine into AleaPoem. Most word processors now include spelling checkers. While you're typing, you can press a key to ask whether you've spelled a word correctly or to check the spelling throughout the document. AleaPoem would have a "meter checker" that worked the same way. Writing or revising a metrical line, you could ask for a scansion of it.

Even this planned version of the editor wouldn't *write poems*. It would automate as much of the poetry-writing process as could be automated. The question I was exploring was, how much of the process might that be? Once I understood that the Scansion Machine could be part of AleaPoem, I knew an answer to that question. I stopped there without doing it—still dreaming.

After all, why go on? The research part was done. As for implementation, I had to ask: If I were successful, what difference would it make? No one who read a poem written with the help of AleaPoem would be able to tell what had gone into it. A poem's meaning is all on its surface, in the sense that if no reader can see it, it doesn't effectively exist. My editor might possibly make some aspects of writing easier, but that in itself is no interesting virtue. I'm a poet, not a software entrepreneur. (And no entrepreneur would bet his shirt on a word processor for poets.) I was growing interested in making a computer make a *difference* in a poem. To do that I needed a new approach.