COMPUTERS IN ENGLISH LANGUAGE EDUCATION AND RESEARCH

NUMBER 1 SEPTEMBER 1985

UNIVERSITY OF LANCASTER
Institute for English Language Education
Department of Linguistics

UNIVERSITE DE PARIS IX-DAUPHINE
Section d'Anglais
I think it is true of many approaches to language teaching that they assume a teacher who is both proficient in the subject matter and intelligent about deciding how to present it, while also assuming a learner who has no proficiency and no intelligence. Under such a model nothing is learned unless it is explicitly taught; learners have to be given, since they cannot take. In this paper I want to speculate about an approach to language teaching which respects the learner's intelligence and to suggest that sometimes an intelligent learner wants and needs an unintelligent partner, a partner who will behave in a totally predictable and rule-governed way.

Some evidence in support of this is offered by Mario Rinvolucri in a recent article describing experiments in adapting computer exercises to ordinary blackboard classrooms, in other words playing computer games without a computer. The exercises, variations of techniques which Tim Johns and I developed, involved reconstructing a masked-out text by trying to guess the words in it, or trying to guess the title of a text from seeing a tiny fragment of it displayed (Rinvolucri, 1984 and personal communication). What Rinvolucri's teachers discovered was that they could not face the class; in a period of sustained guessing, the involuntary signals of approval and disapproval which the teachers transmitted were destroying the exploratory instinct and actually reducing motivation and success. The teachers had to break eye contact and turn their backs on the students. The way I describe the phenomenon is that the teacher facing the class plays the role of GOD, loving, caring, guiding, knowing, chiding. The teachers with their backs to the class were in the role of NATURE, cruel and unforgiving, sometimes appearing arbitrary but in fact governed rigidly by laws of cause and effect.

I suppose that, if you give a presentation a catchy title, you have only yourself to blame if people misunderstand it. In calling this talk Artificial Unintelligence, I wanted to suggest that, while I am aware of the work that computer people are doing under the name Artificial Intelligence, I am more interested in other ways of exploiting computers. Some people have taken my title to imply that I am hostile to artificial intelligence, or
that I intend to make fun of it. Nothing could be farther from my mind. Artificial intelligence is fascinating; it is also great fun. The one misconception about it that I would like to remove is the belief that it is necessary; necessary to us, that is, as educators. Because if we believe that it is necessary, then we are going to have to wait until it delivers some results, and we may have to wait an awful long time.

The trouble is that, if we are going to ask machines to teach for us, then we shall eventually need to make them more intelligent than they now are. As teachers we engage in dialogue with learners, set them tasks and evaluate performance. To do this we have to understand what our students are saying. We also have to understand the personalities and learning strategies of our students so that we can read the feedback signals they send out, not necessarily verbal signals. We need to know if a student is bored, interested, puzzled, tired, or provocative. We must not only know what the learner has said but also why he or she has said it. To equip a machine to engage in tutorial dialogue other than rigidly pre-programmed drill and practice, we must therefore give it a parser which will understand ordinary language, a knowledge base which will allow it to make sense of the language, a model of the learner and a history of each learner's learning style, a means of extending its own knowledge base, and perhaps a sense of humour.

But is such full-frontal teaching what we want the machines to do? Are we in the middle of some crisis of resources that forces us to farm students out to machines? Do we need the machines to do our work for us, even if they could do it better (and on the whole they cannot)? No, the real argument for using computers in language learning is quite different. It is about power. Machines provide us with the power to demand the unreasonable; to demand instant clean copy of a document that we are in the middle of revising, to demand a printout of every occurrence of a particular word in a long document for no better reason than that we are interested to see how it is being used. Machines give us the power to dabble, to experiment, to waste time. One shouldn't waste a teacher's time; teacher's time is an expensive resource, and if one pupil wastes it, the other twenty-nine have cause for complaint. But wasting a computer's time is no sin, since the cheap microcomputer has turned information handling from an economy of scarcity into an economy of plenty. The power of computers can be harnessed by teachers and learners at every level, but only if they understand it. I worry, though, that people who clamour for artificial intelligence solutions to educational problems may not
fully understand the nature of the power the machine gives us.

My term for the kind of teacher I have been talking about, the kind who acts as an initiator and controller of events, is magister. The archetypal magister is qualified, receives a salary, and has authority to approve and to punish. There is another possible model of an educational functionary, that of pedagogue, which is the classical term for the slave who escorted the children to school. Serving mainly as a bodyguard, the pedagogue would also answer questions, translate words, help with homework, or even administer tests (though only on demand). There are not, thank goodness, very many human pedagogues around nowadays. We cannot afford them, and we have a guilty conscience about treating humans in such a degrading fashion. But the pedagogue function is the natural one to give the machine. Computers were originally designed to answer questions, not to ask them. They are responsive devices, with no desire to talk to us, to initiate conversation. Such devices can be of enormous use provided that we take on the responsibility of initiating, provided that we get into the habit of asking lots of questions. That is when the machines come into their own.

Perhaps the most neglected role of the ones we can readily give to computers is that of demonstrator. This first occurred to me five years ago when I was writing my very first BASIC program, a little exercise on word order. The screen displayed four animals, elephant, crocodile, cat and mouse, crudely drawn in character graphics, and assembled a question of the form WHAT HAS THE CAT EATEN? or WHAT HAS EATEN THE CAT? The user would answer THE MOUSE to the first question or THE CROCODILE with the second type of question. The answers to WHAT HAS EATEN THE ELEPHANT? and WHAT HAS THE MOUSE EATEN? would be NOTHING. As I was putting this together, I said to myself, let’s have ten items in the exercise, and we had better have two examples. Then I stopped to ask myself, why two examples? Why not three? Or three hundred? Since the machine could generate examples randomly for ever, why should I decide in advance when the user should switch from observing to doing?

For learners often want to be receptive, want relief from being challenged and potentially humiliated. I was made very aware of this during my recent and very unsuccessful attempts to learn some Serbo-Croatian. My private teacher before my departure, who should have been my pedagogue but clearly saw herself as magister, thought she knew what I knew and what I needed; since I pronounced
the sentences well and quite often (thanks to luck and logic) got the answers to her questions right, she was convinced that I understood everything I was saying and had mastered the underlying language, so she rushed me into more advanced material, virtually covering a hundred hour syllabus in ten hours. I kept pleading with her, "No, I don’t know that yet, give me some drill", but she did not listen. The result was that I retained no durable grasp of the language. With a machine to give me the demonstration and drill I craved at the time, I might have done rather better.

Does this mean that I am suggesting we use computers for drill and practice? Yes, but with an important reservation. Drill as a service, drill on demand but only on demand, is a very different entity from drill which is structured and imposed by an outside agency. Drill itself can become a form of linguistic play, and play is the missing element in much organised learning, an element the computer can give back to us. And to a great extent what I am proposing we should do we is not for the machine to drill the learner: rather it is the learner who will drill the machine.

The kind of help I wanted from my Serbo-Croatian teacher can be illustrated with a program produced during a programming course nearly two years later when I realised that I still did not know the days of the week. How nice it would be to have a slave to tell me what I wanted to know in context on demand, and then test me if, but only if, I wanted to be tested. So I asked one of my Serbian students to help me write the program I call SERBDAYS, and this is what we produced.

```
M T W T F S S
o u e h r a u
n e d u i t n

Danaj je petak.
Sutra je subota.
Juce je bio cetvrtak.

L=left     M=menu     R=right
```
The user points the arrow at a day, and the three relevant sentences (Today is .../ Tomorrow is .../ Yesterday was...) come up in the box. One can spend as long as one likes just shifting the arrow and looking at what comes up. The program also incorporates a testing phase in which the word for the relevant day in the first sentence is blanked out and has to be typed in. The unusual thing about this phase is that the input is controlled so that only the correct letter is accepted. If you press something else nothing happens at all; no bells ring; no little messages appear telling you what a bad girl or boy you are. Similarly no congratulations appear when you get it right. If I employ a slave I do not want the slave to criticise me, nor to congratulate me. I just want the slave to obey orders and to report facts. I do not want my computer to be user-friendly or, to put it another way, I do not want my slave to be master-friendly.

The demonstrator role extends into that of playmate or stooge in the kind of program in which we ask the machine to produce language randomly. We then look at what it produces, laughing at what is ridiculous, pondering over what is accidentally profound. The best known form for such programs to take is the poetry generator, but I always feel disappointed that so much attention goes to free-form poetry when the machine can very readily produce simpler and more everyday language. All it is doing, after all, is making selections from a substitution table, and these can take the form of conversation, as in Burkhard Leuschner's DIALOG-CRIT, of zany narratives as in the game of MADLIB, or of anything else. My wife and I have been working on a suite of such programs which she has christened NONSEQUENCES. The machine assembles new proverbs, original advice for tourists in London, pithy proverbial Scottish wisdom, or samples of reported conversations. It could handle a great variety of sentence or discourse types, anything indeed that was on a structural syllabus, but the mode of use is rather closer to what Earl Stevick asks us to do with structural material than what the traditional teacher does with it (see, eg, Stevick, 1982, ch. 9). The machine simply churns out samples. We, the learners, look at what it produces and decide what we like and want to preserve on the printer, rejecting anything which is dull or simply inconsequential. With the PROVERBS component, I have run competitive sessions, with groups of learners trying to assemble the ten best new proverbs, submitting their collection to the vote of the whole class. Here is a sample of the sort of printout the program gives.

Roadworks promise frustration.
Roadworks are worse than a boring seminar.
The Olympic Games are worse than a coincidence.
Love and marriage are like the weather forecast.
The next stage, of course, is to ask learners to contribute new components, to extend the data list. When they do this, they may well find that what they have put in leads to ungrammatical language (the group will quickly spot this even if an individual does not). This may give them insights into the grammar or semantics of the piece of language, which they might not have got so easily from a teacher’s correction of their own mistakes.

Even though the programs I am talking about synthesise language, I doubt whether anyone would describe them as artificially intelligent. The process of synthesis is entirely rule-bound, and the machine has no means of 'understanding' what it is 'saying'. Yet there is intelligence present during the interaction, the learner’s intelligence in assessing, responding to, criticising, or enjoying what the machine sets up, and it seems that the learners’ recognition of the machine’s stupidity is a factor in releasing their own intelligence and zest for experiment.

One gets a little closer to the domain of artificial intelligence if one designs programs in which the communication is more genuinely two-way, where the learner has to get messages across to the machine as well as receiving messages from it. This means equipping the machine with a parser. The parser, however, does not need to be perfect provided that we have not endowed the machine with an aura of omniscience. Users will be perfectly ready to modify their language and try other formulations to find ones which work as long as they see the machine as basically stupid. This is already an observed factor in the educational use of adventure games; Daniel Chandler, for instance, has reported real learning benefits among young first language learners who have to learn to reduce their input to two word commands when they play with adventures (Chandler, 1982). In the process they start thinking about the nature of communication in a new way; they realise that they have to take the listener or reader into account when they speak or write.

My current work is concerned with devising very simple parsers to cope with user input to logic games and other kinds of exploratory program. One of these is for a program called TIGLET in which a tiger cub asks for food, and the user must offer it different things to eat. The program works with a classified vocabulary in which every kind of food belongs to one major type, such as meat or dairy product, is associated with one or more adjectives, such as sour or expensive, and is also associated with its typical colours. At the beginning of the game TIGLET chooses a category, an adjective and a colour. He will
answer 'I quite like that' if the food matches on one count, 'I like that' if it matches on two and 'I love that' if it matches all three, 'I don't like that' if he cannot find a match, and 'I have never tasted that' if the user inputs a food outside the known vocabulary. The user is trying to guess the three categories; interestingly enough by far the most useful information comes from what TIGLET refuses rather than from what TIGLET likes. But the learner is also trying to communicate in a subset of natural English which TIGLET has been 'taught' through the parser. It is a very limited brute force parser, needing only to cope with about three hundred words, and expecting nearly every sentence to be an offer, which greatly simplifies the semantics. The limitations of the parser, the fact that it does not cope with every possibly way of making an offer, need not be a drawback, provided that it can handle notions of quantity and count v. non-count; this is the problem which is at the front of my mind at the moment. There is more to TIGLET, however, than just the logic game; one can also teach TIGLET new vocabulary and therefore new ways of classifying food. This, I expect, will be the most engrossing part of the activity, since it will force learners to think about the consequences of description. Is a tomato a fruit or a vegetable or both? What colour is an apple? Should one include brown, since rotten apples go brown? TIGLET won't help; he just waits to be taught. Whatever the learners are learning is having to be learned from experience or induced from the output they eventually get from the program.

If one wants to classify acts of reasoning and learning into deduction and induction, i.e. moving from general to particular or from particular to general, then there is little doubt that TIGLET and his kind belong to the inductive approach. TIGLET supplies examples and data, but it is the user who has to create the larger classifications and principles that will make sense of them. Learning can, usually does, involve both deductive and inductive procedures, but styles of teaching usually show a preference for one or the other. I think there can be little doubt that a strongly inductive style of teaching (notice I do not say learning) is rather rare, though one could perhaps find it in the Silent Way, in TPR or in a whole-hearted application of Krashen's Natural Approach. Such inductive approaches, however, share the characteristic of being carried out in an unaware fashion; the learner is being encouraged to think about the meanings to be conveyed and not about the means of conveying them. The kind of exploratory and problem-solving approach which I have been describing differs by encouraging reflection about the means, an overt analysis of language itself.
This in some ways parallels the difference between the two major deductive approaches: structural or drill-based on the one hand, promoting habit formation while suppressing conscious analysis of language, and cognitive code on the other, allowing attention to be given to the means as well as to the end. We could display this classification as a grid:

<table>
<thead>
<tr>
<th></th>
<th>Deductive</th>
<th>Inductive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unaware</td>
<td>Pattern-practice</td>
<td>Immersion</td>
</tr>
<tr>
<td>Aware</td>
<td>Cognitive code</td>
<td>Exploratory</td>
</tr>
</tbody>
</table>

The explicit and aware use of induction is not common in language teaching, except perhaps in the Silent Way. Perhaps we could relate this to the other element missing from much formal tuition but very common in infant learning, conscious linguistic play. Play, however, needs a playmate, a mother, a sibling, or a friend. Teachers are not really fitted for this role; they are too intelligent and take too many initiatives. They bend the rules of the game, often in a well-meaning effort to help.

But that is something the computer will not do. It follows the rules. It is too stupid to do anything else. What we have to realise is that its very stupidity can be turned into an asset, since it releases the learner’s intelligence, the learner’s hunger for self-knowledge, and the instinct to explore. Until we know that we can make demands on a slave, however unreasonable, we will refrain from demanding enough. What the machine can do for us is turn language learning into an experimental subject, a subject where the learner tries things out, measures the effect of linguistic choices, and derives perceptions and insights by making sense of authentic data, data which the machine can organise. Just as chemistry students have their chemistry laboratory, so the computer can provide us with a language laboratory, namely an experimental environment for language learning. What we currently call a language laboratory is nothing of the kind, since no experiments occur in it; regardless of what you say into the microphone, the tape will give you its pre-recorded response. Computers, in contrast, facilitate and encourage experiment. I would hesitate to say that they are going to make language learning easier, but I am sure that they will make it more exciting.
References:

