Constructivism, Self-Directed Learning and Case-Based Reasoners: A Winning Combination

P. Boylan
Dipartimento di Linguistica
Università degli Studi "Roma Tre"
Via Castro Pretorio, 20
00195 Roma, Italy
boylan@uniroma3.it

A. Micarelli, V. Pirrottina and **F. Sciarrone**
Dipartimento di Informatica e Automazione
Laboratorio di Intelligenza Artificiale
Università degli Studi "Roma Tre"
Via della Vasca Navale, 79
00146 Roma, Italy
micarel@dia.uniroma3.it

Abstract

The BLITS system described in this paper is a case-based system designed to help users draft effective business letters in English. The typical end user has at least a secondary school education, is computer literate, and knows the English one learns at school or university (how to write correctly, but not how to write effectively). The learning support offered by BLITS is based on two modern educational approaches: constructivism and self-directed learning. BLITS does not teach "rules" for writing effectively and does not offer step by step training in converting ideas and intents into a well-written letter. On the contrary, the system leaves it up to users to make judgments of what effective writing means in a given situation, taking into consideration a number of suggested alternatives and the results those texts obtained in previous correspondence. It is therefore the development of the learner's judgment – the capacity to choose words judiciously instead of applying rote formulas – that makes working with BLITS, like all constructivist-inspired learning aids, a truly educational experience.

Introduction

Most intelligent learning support systems, in particular Intelligent Tutoring Systems (ITS), assume that "knowledge" is a predefined entity. This assumption applies not only to the knowledge that humanity possesses in a particular domain (e.g., our knowledge of mathematics) but also the knowledge that particular individuals must have in order to handle given tasks or understand productively a given domain. Thus an ITS typically has a knowledge base consisting, for example, of the concepts and procedures that high school students are supposed to need in order to carry out routine computational tasks or that math majors are supposed to need in order to demonstrate a theorem. The ITS interacts with users in such a way that the "needed" concepts and procedures become operative in them.

What this comes down to, then, is a fundamentally mechanical model of knowledge transmission. No matter how open-ended and adaptive a typical ITS may be, what is inside the system is meant to end up inside the heads of the students. Just how it gets there and how it is formulated may vary from student to student[1] but the bottom line remains the same: the ITS, like Big Brother, knows what the student is to know and that is what is learned.

This fundamentally dogmatic vision of knowledge has been questioned by two modern currents in educational thought:

- the constructivist view of knowledge as fundamentally unpredictable in scope and extension (Jonassen, Mayers, & McAlesee 1997); precursors are (Piaget 1923) and (Vygotsky 1956);

- the educational movement called "self-directed learning" which developed in Great Britain and elsewhere in the 1970's (Trimm 1976); precursors are (Dewey 1996) and (Montessori 1935).

Constructivism is a well-known and well-documented current (see the bibliography in Jonassen, Mayers, & McAlesee 1997). Self-directed learning is less so, especially in the United States where it is often confused with "teach yourself" books and computer-assisted instruction (CAI).

But self-directed learners are not simply people who learn, on their own, the educational content of a book or computer program. Indeed, self-directed learners avoid books or computer programs that tell them what they supposedly need to know. Instead, they choose resource materials that can be pieced together to fit the educational goals they set for themselves. Self-directed learners also choose the criteria by which they may judge when they have attained their learning objectives.

They key difference, then, between "autonomous" and "self-directed" learners is that the latter take charge of their own intellectual development, determining what kind of "knowledge" they want, how much of it they need, and how to tell when they have attained it (Trimm 1976). In general, their knowledge goals are specified in results-oriented, contextualized, behavioral terms: "I will be able to affirm that I know how to play the guitar when I manage to strum a tune at parties" or "when people think they're hearing Eric Clapton" or "when I manage to play stuff that I, at least, appreciate" or whatever.

[1] User Models can in fact be extremely flexible (Rich 1983).

But does learning of this kind work with more "serious" disciplines, such as those taught in institutions of higher education? We believe so and, to test our hypothesis, we have developed an educational system that teaches the rhetoric of successful letter writing in the business world.

Case-Based Education

Just as "educating" does not necessarily mean getting students to assimilate predefined "knowledge", "computing" does not necessarily mean implementing process-control algorithms. Although such algorithms form the bulk of most programs, there are other "less directive" computational models to choose from.

Case-Based Reasoners (Schank 1990) (Aamodt & Plaza 1994), for example, offer their users resources with which to elaborate original views of the world. The "knowledge base" that the system builds up over time and the "knowledge" that the user acquires are to a large extent freely constituted and unpredictable. CREANIMATE, for example, is Schank and Edelson's program to teach children how animal organs function by permitting users to create imaginary beasts made up of the organs of different kinds of real animals and then to test their hypotheses. While the system makes use of a predefined knowledge base consisting of "cases" of how of particular organs work (for example, how wings and fins are used to navigate), the user may create new animals and insert them into the "knowledge base". This extends her knowledge of organ functionality in hypothetical worlds. The user's inventiveness is limited by physical laws, of course, which is what she discovers as she tries to make her imaginary animals ever more efficient. But what she ends up knowing is up to her; it can include domains foreseen by the system (for example, "kind of protuberance" vs. "kind of navigation") or domains that are original ("the aesthetics of protuberances", i.e. the student has used the system to create realistic and functional beasts for dungeon video games).

The Business Letter Composition system we have developed uses a Case-Based Reasoner to implement the educational philosophy just sketched. Thus, the system does NOT teach the "rules" of rhetoric and does not even contain "lessons". It lets the user work out her own rules of rhetorically effective discourse by trying to write letters while taking inspiration from cases of good letter writing that the system displays. If the System is able to produce, just when they are needed, examples of the kind of letter that the user is looking for, appropriately tagged to show rhetorical development, learning will take place. The user will inductively acquire knowledge of the rules of good rhetoric and will end up being able to produce effective letters without the help of the system.

The System

Our prototype system is targeted to adult users whose job entails writing business letters in English, either for themselves (Small Office and Home Office users) or for their superiors (secretaries in large offices). These users do not normally have an "ear" for good writing. This is all the more true if

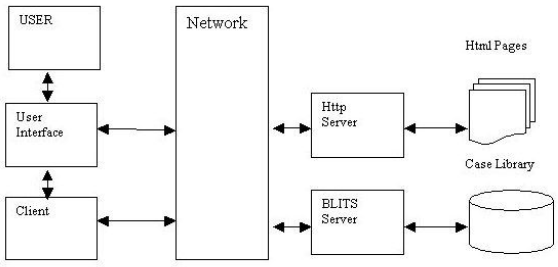

Figure 1: The Architecture of the System.

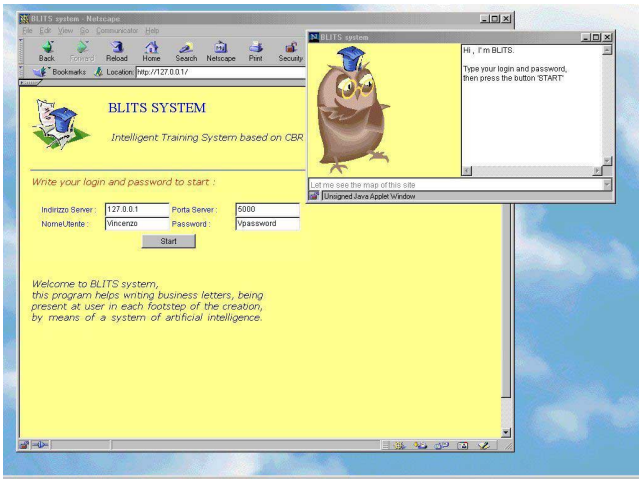

Figure 2: The Login Page.

they are not native speakers of English. They cannot, therefore, rely on their own untrained judgement in deciding the best way to request, complain, offer, threaten, etc. What these writers normally do is to find letters to act as models – initially from a Guide to Good Correspondence and then from past correspondence that has proven to be effective – and then cut and paste suitable parts of the old letters to form a new one.

Our system helps them do these very same steps, but more intelligently. Through composing letter after letter, users grasp what effective letter writing entails IN THEIR OWN TERMS and ACCORDING TO THEIR OWN NOTION of what "good writing in English" means. For example, "good written English" means one thing in Britain, another thing in the U.S., and something else in Hong Kong, where SouthEast Asian "Business English" thrives and where using "proper English" (which smacks of colonialism) is rhetorically less effective in business correspondence (Crystal 1997).

BLITS is implemented in a client-server architecture and consists of four modules, as shown in Fig. 1. The first is the User Interface Module, responsible for the direct user-system interaction. It includes an intuitive interface, developed by means of html pages and Java applets, accessible through a browser. The second module is the Client Mod-

Figure 3: The Option Page.

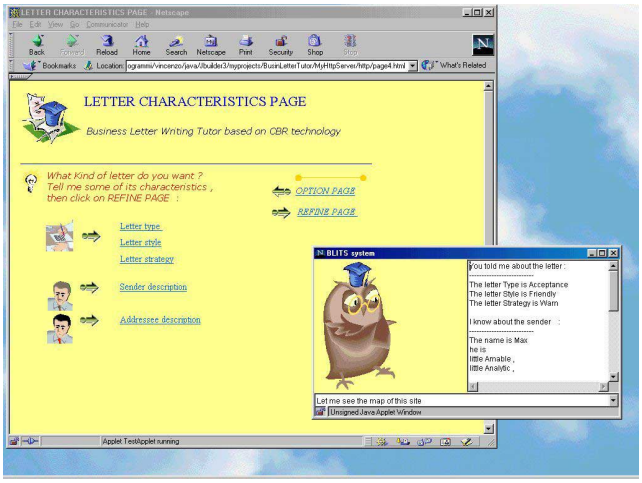

Figure 4: The Letter Characteristics Page.

ule, which communicates with the Server Module through Internet, by using the TCP/IP protocol. It is written entirely in Java. The Server Module processes the requests from the different clients it is linked to and supplies to each the requested information. This module is also written in Java. Finally, the Case Library Module is a dynamic library which stores the information gathered from cases of letter writing solved in previous interaction with the system and tagged with respect to their effectiveness in real-life interaction. The next section will describe the User Interface Module by means of an example of a simple interactive session with the system.

An Example Session

The first page displayed to the user (see Fig. 2), offers a brief introduction to the BLITS system and enables login. Once users are authorized to continue with the use of the system, the "Option Page", shown in Fig. 3, is displayed, where the learning assistant – represented by a small "owl" – offers the user three choices: (i) write a new letter, (ii) retrieve an old letter and (iii) indicate the effectiveness of a previous letter. If the user clicks on "Write New Letter", s/he is transferred to the "Letter Characteristics Page" as shown in Fig. 4. On the left are listed the desired characteristics of the letter: type, style and strategy. Below them appear the characteristics of the recipient and the sender. During the construction of the letter to send, the user may have to return to this page; when s/he does, the assistant will present a summary of the choices made up to that point.

Users are not forced to specify the values of all the characteristics necessary for the new letter. Clearly, however, the accuracy of the system in proposing suitable model paragraphs increases in proportion to the amount of information given.For example, by clicking on "Sender Description", the sender's characteristics can be indicated: see the page shown in Fig. 5. The sender can be the user or another person, for example, the boss. If a letter has already been written by the same sender, it is possible to select it directly from the pop up menu. If the sender is a new person, the new name must be inserted in the appropriate box and a brief questionnaire must be completed giving the sender's characteristics. A click on the "Letter Characteristics Page" will prompt a return to the preceding page and the same operation is repeated for the recipient. After these preliminaries, the user clicks on the "Refine Page" button and is transferred to a page enabling her/him to refine her/his previous selections. After the retouching phase (which can be skipped), the user clicks on "Send to Server" and waits briefly for the necessary data to be downloaded. The data consists of a list of possible communicative intents and an initial series of cases suited to the specific letter-writing situation.

After the Client Module downloads the requested data from the server, a "Move Selection Page" appears (see Fig. 6). This page enables to user to select the "moves" (as in a chess game) s/he wishes to make to attain the goal s/he has in mind. We call that goal the user's communicative intent; it is never explicitly defined by the system and is probably more or less unconscious in the user's mind as well. We define it as the "sum of the chosen moves" or, as it is represented in the system, the "path connecting a series of chosen nodes". (This lack of explicitness may seem strange. It is like defining the goal of a diver with a long sentence – "jumping up in the air, touching one's toes, elongating one's body head down, plunging into the water" – instead of using a simple term from a diving repertory, e.g. "jackknife". This way of working corresponds to the concrete character of knowledge typical of constructivism.)

Let us imagine that the user selects "Acknowledge Receipt" as her/his first move and "Apologize" as the second move (see the menu offered on the left of the figure). The CBR engine – which is also present on the client machine – automatically re-orders the examples (shown in the lower section) and displays, in decreasing order of usefulness, the most suitable candidates for the next move to make. Let us assume that the user chooses 'Refer to Past Behavior" as her/his third and final move (see the figure). Once the

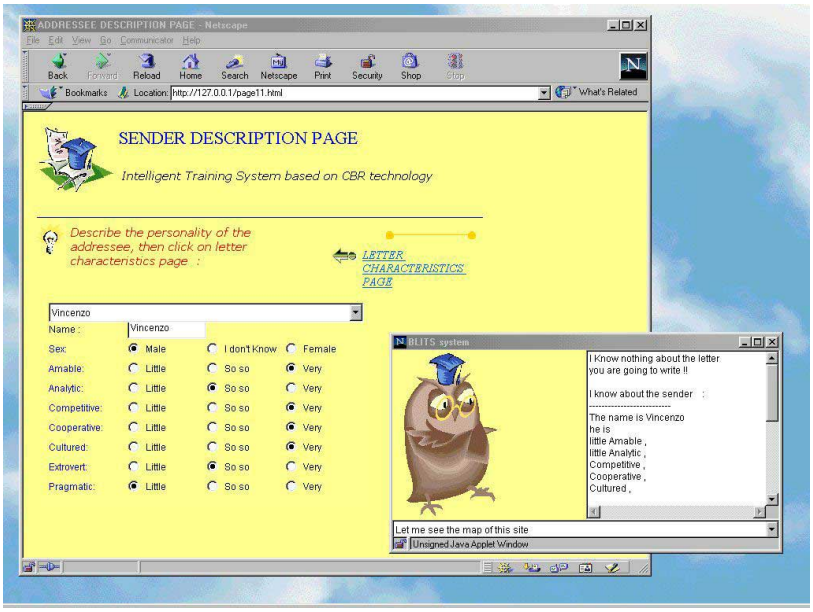

Figure 5: The Sender Description Page.

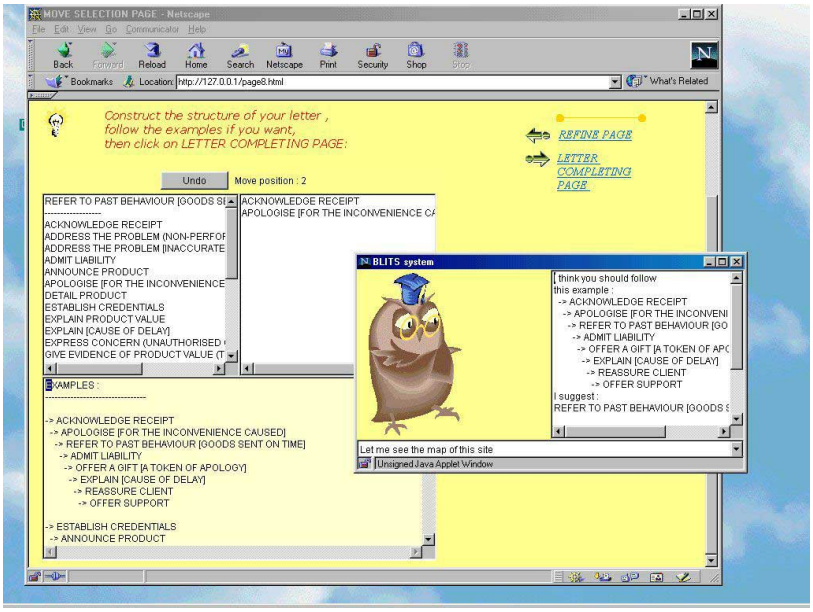

Figure 6: The Move Selection Page.

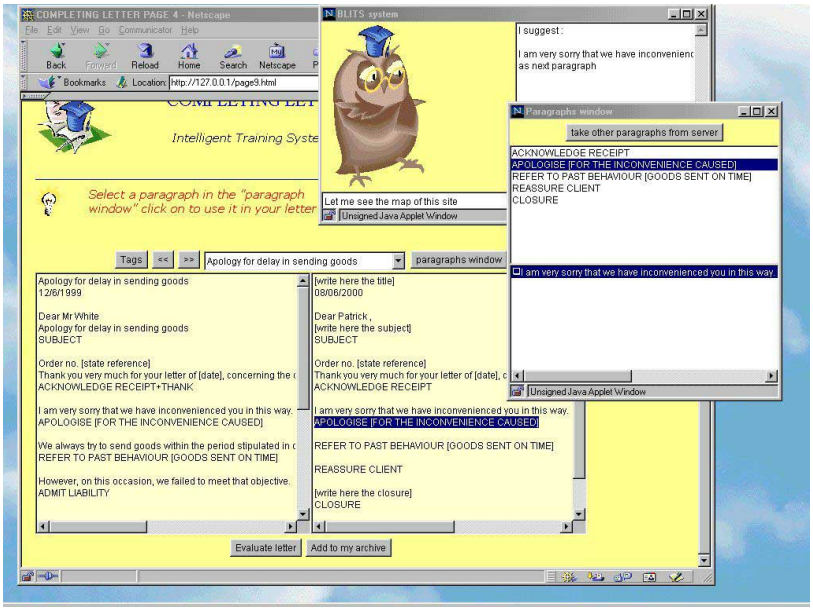

Figure 7: The Completing Letter Page.

sequence of moves has been chosen, the user clicks on the link “Completing Letter Page” and is transferred to the page shown in Fig. 7. It enables the user to begin the composition of her/his letter by selecting one paragraph for each of the moves s/he has decided to make.

Some sample letters (the cases retrieved from the server Case Library) are listed on the left-hand side of the screen. By activating the pop-up menu, it is possible to decide which of these to view in the text window on the left. The text window on the right serves instead for drafting the new letter, which can be typed out directly by the user or composed more rapidly by choosing from model paragraphs presented in the “Paragraph Window”. The latter is divided into two parts: the upper portion includes the moves selected in the preceding section, while the lower portion displays the paragraphs that can be copied onto the new (draft) letter. Let us imagine that the user clicks on “Acknowledge Receipt”, i.e. the first of the three moves that s/he has previously chosen as the backbone her/his letter. This click will cause a number of paragraphs to appear, each one being a realization of that particular move, ordered according to their suitability for the current letter-writing situation. Let us then imagine that the user chooses the first paragraph displayed by clicking on it. That paragraph will be automatically copied into the window on the right where it will constitute the first paragraph of the new (draft) letter. The user can retouch it or leave it as it is. Let us imagine that the user prefers to leave retouching to the end and moves on to choose a paragraph for her/his second move, “Apologize”. By clicking on the move and then on one of the paragraphs that the system presents as a realization of that move, the user adds a second paragraph to the draft letter in the window on the right. After adding the third paragraph and doing the necessary retouching with the help of the Owl (this part of the program has yet to be implemented), the user is ready to press the “Evaluate Letter” button and, finally, the “Add to my Archive” button. The “Evaluate Letter” command directs the Owl to provide a judgement on the letter that has just been drafted on the basis of resemblance with the Letter Stereotypes (prepared by an expert and included with the system) and the Past Cases of letters effectively written, sent, and judged subsequently by the user on the basis of the results obtained in the real world (i.e., the boss’s smile, a client’s favorable reaction, etc.). If the Owl indicates that the letter, as composed, is likely to prove unsatisfactory, the user can decide to keep it anyway, send it, and indicate (in a future session) full satisfaction with the results it has obtained. This action will update the weighting system that permits the Owl to furnish judgments about letter suitability. Or, if the user so chooses, s/he can accept the Owl’s criticism, return to the “Move Selection Page”, reassess whether the moves s/he chose really accomplish her/his communicative intent, and re-do the paragraph selection operation. In doing her reassessment s/he is helped by the non directive comments that the Owl furnished together with the negative assessment.

Conclusions

At first glance, the non-directive manner in which the user composes a letter may seem scarcely educational. In other words, users may appear to be left to fend too much on their own, causing feelings of frustration. Indeed, a critic might say that the BLITS system resembles the “grammar checkers” furnished with leading word processing programs. Although these programs try to be prescriptive, saying what is right and what is wrong, they often are unable to decide on the value of a word in a given context and therefore end up inviting the user to reflect on whether the word is really appropriate or not. This, the critic would add, makes these

grammar-checkers marvelous non-directive educational aids (according to the philosophy defended by this paper) precisely when they work least well. Moreover, the critic might add, people tend to stop using these grammar-checkers precisely when they leave too much to the user's judgment: users do not want to solve problems, they want quick answers.

These two objections are quite reasonable. As for the first, we would answer that, yes, grammar-checkers are indeed the most educational when they work least well (i.e., when they don't furnish quick answers) and that is precisely what our system aims to do. But while grammar-checkers irritate and leave users perplexed by the absurdities they sometimes come up with, BLITS compensates the user's patience by offering (hopefully) highly appropriate alternatives. The grammar-checkers described are not educational because they are, too often, like talking to a moron; using BLITS (hopefully) is like engaging in an intelligent conversation. The educational help BLITS offers, as with self-directed constructivist educational aids in general, therefore depends on the quality of the creative thinking it engages the user in. If the cases furnished are truly "to the point", they will stimulate the user's intelligence. If they are completely "off base", they will engender frustration, just like the grammar-checkers cited. So the real issue is how well the CBR algorithms work in coming up with cases that fit the situation at hand and, consequently, with pertinent advice during the Evaluation phase. That issue has been discussed in a previous paper (Papagni *et al.* 1997). We feel there is room to improve our CBR model but that we are on the right track: people do reason in cases and knowledge is indeed a sedimentation of past cases and this is precisely what CBR technology is meant to capture.

The second part of the objection is harder to answer. Yes, it is true that people want quick answers ("the big red button" to push). Learning may be stimulating, but if you are a harried Italian secretary in a noisy office with an irritable boss who wants an effective letter of apology written in perfect English before lunch, you will not be in the right frame of mind to indulge in a thoughtful educational experience lasting a quarter of an hour. We have tried to meet this objection by providing our system with a series of back doors through which to leave when one is in a hurry: the user can recall a past letter in its entirely instead of composing one, or, while composing, can check the default version initially furnished, or if s/he decides to create the letter paragraph by paragraph, s/he can skip the tutorials (to be implemented) that accompany the retouching phase, doing the tidying up as s/he would if s/he wrote a letter from scratch using past correspondence as a model. In other words, the user can opt for as much instruction as s/he has time for. Self-directed education does not try to turn every learner into a Hemingway, as prescriptive/directive educational systems do. If a user is happy with faulty, barely satisfactory letters because s/he has no time or interest to learn to write better, s/he has the right to choose this educational goal and use her/his energies in other, more (subjectively) worthwhile activities. The environment (the boss's dissatisfaction, the clients' ironical comments) will exercise enough pressure on the user to make sure she dedicates at least a minimum amount of time in using the system's capabilities.

Besides, what results do prescriptive/directive educational systems obtain in practice? It is notorious that lockstep programs that drill users into using "good writing" rules are seldom purchased and, if they are, become little used. How much educational value does that kind of educational aid give? Zero. So perhaps our "less" is indeed more.

The question, as one can see, is vast and touches the very philosophy of our educational system in general. Given the dropout levels of present-day schooling and the poor results of many of the students who remain, the failure of prescriptive/directive teaching is, if anything, a fact. Perhaps it is time to risk trying another educational model. That is what we are attempting to do, together with the CBR community at large, in implementing the system just described.

References

Aamodt, A., and Plaza, E. 1994. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AICOM* 7(1):39–59.

Crystal, D. 1997. *English as a Global Language*. Cambridge: Cambridge University Press.

Dewey, J. 1996. Collected works. In Hickman, L., ed., *Valuation and Experimental Knowledge*. Charlottesville: InteLex Corp. (orig. 1922).

Jonassen, D.; Mayers, T.; and McAlesee, R. 1997. A manifesto for a constructivist approach to technology in higher education. In Duffy, T.; Jonassen, D.; and Lowyck, J., eds., *Designing Constructivist Learning Environments.* Heidelberg: Springer-Verlag.

Montessori. 1935. *Manuale di Pedagogia Scientifica*. Napoli: Alberto Morano Editore.

Papagni, M.; Cirillo, V.; Micarelli, A.; and Boylan, P. 1997. Teaching through case-based reasoning: An its engine applied to business communication. In du Boulay, B., and Mizoguchi, R., eds., *Proc. of the World Conference on Artificial Intelligence in Education AI-ED 97, August 18–22*, 111–118. Kobe, Japan: IOS Press.

Piaget. 1923. *La Language et la Pensée chez l'Enfant*. Paris: Delachaux & Niestlé.

Rich, E. 1983. Users are individuals: Individualizing user models. *The International Journal of Man-Machine Studies* 18:199–214.

Schank, R. 1990. Case-based teaching: Four experiences in educational software design. *Interactive Learning Environments* 1:221–235.

Trimm, J. 1976. Some possibilities and limitations of learning autonomy. Technical report, Department of Linguistics, University of Cambridge.

Vygotsky, L. 1956. Mind and language. In Leont'ev, A., and Lurija, A., eds., *Izbrannyie psichologiceskij isslédovaniia*. Academy of Pedagogical Sciences, Moscow.