Using Production to Assess Learning: An ILE That Fosters Self-Regulated Learning

Philippe Dessus, Benoit Lemaire

Using Production to Assess Learning: an ILE that Fosters Self-Regulated Learning

Philippe Dessus and Benoît Lemaire

LSE

Université Pierre-Mendès-France

1251 avenue centrale, BP 47

38040 Grenoble Cedex 9, France

Phone: (33) 4 76 82 57 09 **Fax**: (33) 4 76 82 78 11

Email: {Philippe.Dessus, Benoit.Lemaire}@upmf-grenoble.fr

Abstract: Current systems aiming at engaging students in Self-Regulated Learning processes are often prompt-based and domain-dependent. Such metacognitive prompts are either difficult to interpret for novices or ignored by experts. Although domain-dependence per se cannot be considered as a drawback, it is often due to a rigid structure which prevents from moving to another domain. We detail here a system that addresses these limitations. This two-loop system provides texts to be learned through summarization. The first loop is called *Reading*, in which the student formulates a query and is provided with texts related to this query. Then the student judges whether each text presented could be summarized. The second loop is called *Writing*, in which the student writes out a summary of the texts, then gets an assessment from the system. In order to automatically perform various comprehension-centered tasks (i.e., texts that match queries, assessment of summaries), our system uses LSA (Latent Semantic Analysis), a tool devised for the semantic comparison of texts.

Key-words: learning environment, architectures

Introduction

This paper is about Lesgold's "two fundamental laws of instruction" (Lesgold, 1988, p. 118): (1) "Not everyone who passes a test on a topic knows what appears to have been tested"; and (2) "Not everyone who fails a test on a topic lacks the knowledge that appears to have been tested". Referring to these two laws, it is worth studying the means to reduce the gap between student knowledge and its assessment by a system.

Apex, the system we are developing, aims at decreasing this gap by comparing the texts judged to be learned by the student to their summary. The architecture of our system is composed of a main loop, divided into two components: one devoted to reading, the other one devoted to writing:

? *In the Reading component*, the student reads texts and, after each one, is asked a judgment of learning. The first text is selected according to a natural language query provided by the student. Other texts are successively selected by the system according to the student judgment of learning of the prior ones.

? *In the Writing component*, the student writes out a summary of what has been read. The system compares this summary to the texts the student has judged as being learned. The student is warned in case the self-assessment is not coherent with the assessment by the system. The summary is then revised as often as the student wishes.

In order to automatically perform these comprehension-centered tasks (i.e., texts that match queries, assessment of summaries), our system relies on LSA (Latent Semantic Analysis), a tool devised for the semantic comparison of texts that we will describe later.

The features described above exist in the current research literature within two lines of research: ILE, for Interactive Learning Environments and SRL, for Self-Regulated Learning.

Apex = ILE + SRL

To date, systems called ILE are either *production-centered*, qualitatively assessing production to help students write texts (Boylan & Vergaro, 1999; Zellermayer, Salomon, Globerson, & Givon, 1991) or solve problems (Veermans, de Jong, & van Joolingen, 2000); or *comprehension-centered* (Aist, 2000; Salomon, Globerson, & Guterman, 1989), providing students with texts related to their comprehension of previously delivered texts.

The main features of ILEs are the following (Self, 1994): (a) *students are autonomous*, the guidance of ILEs is less formal than traditional ITS, (b) *focus on non-domain knowledge*, ILEs are more domain-independent than traditional ITS, and they tend to foster students' metacognitive skills, (c) *vary teaching styles*, since knowledge transmission is not the core goal of ILEs, they adopt various teaching styles (i.e., dialogue, assessment,

scaffolding, and so on), (d) *authentic learning*, situations presented in ILEs are more authentic, giving students opportunities to develop real-life skills (i.e., writing and revising summaries, self-assessment of understanding, and so on).

We combine this notion with another line of research: Self-Regulated Learning (SRL), which commonly relies on four assumptions (Azevedo, 2001): (a) the learner is an active, constructive participant in the learning process, (b) learners can monitor, control and regulate some aspects of their own cognition, motivation and behavior, (c) there are some goals or standards the learner can set in order to carry on with the learning process, (d) self-regulated activities are mediators between learners and performance (e.g., reading or writing).

Few research has been done to promote SRL within ITS. One of the reasons could be that "computer-assisted SRL" is an oxymoron: "self-regulated" means that the student is free to perform a given activity, while the very notion of tutoring means that the computer constraints more or less formally the student activity.

A way to address this drawback is to integrate the SRL theory within an ILE rather than in an ITS. We will first describe some trends about SRL research, then we will present LSA. Finally, we will detail our system, *Apex 2.0*.

Self-regulated learning: theoretical framework

We adapt the model of Self-Regulated Learning, detailed in Butler and Winne (1995). These authors describe their model as follows (see Figure 1): first, the student constructs a representation of the learning task through his or her prior knowledge (e.g., beliefs, knowledge on domain and strategy, motivational beliefs). This representation is used to set learning goals, that are progressively attained by diverse tactics and strategies. Then these goals generate products, both internal (cognitive and emotional) and external (behavior, performance). This last phase is subject to two sorts of metacognitive processes: (a) it is first *monitored* by the student, who can reassess goals and change his or her strategies through internal feedback; (b) then, performance can be cognitively *controled* (Nelson, 1999). In other words, learning occurs when the teacher helps students to formulate and monitor their own goal instead of doing it in their place, from an expert point of view. We now describe systems devoted to assist this goal formulation in various academic activities, and therefore to foster SRL.

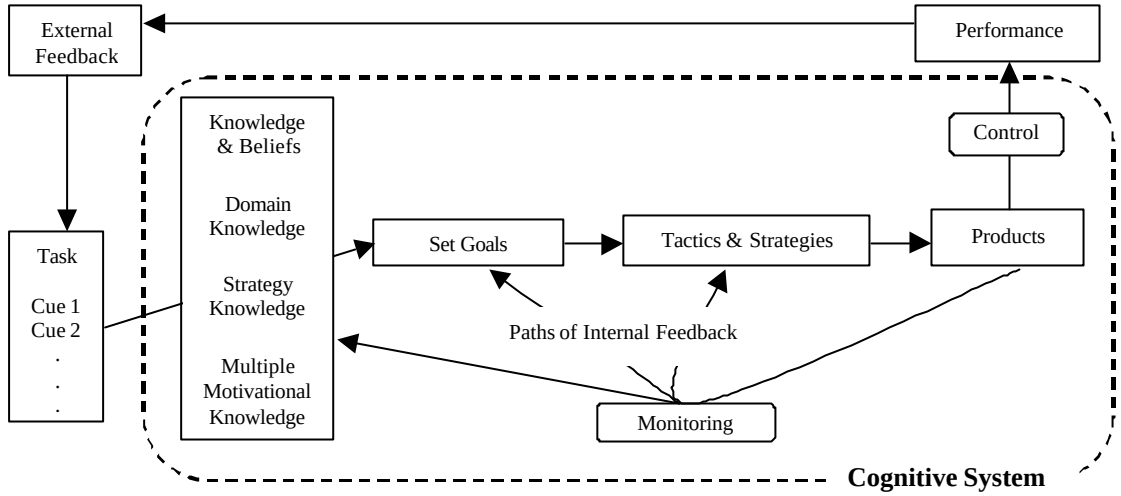

Figure 1 – A Model of SRL (after Butler & Winne, 1995, p. 248; Nelson, 1999)

Related work on computer-aided SRL

One of the first SRL-centered computer programs was *Reading Partner*, developed by Salomon et al. (1989). It delivers metacognitive questions during reading. *Reading Partner* includes neither a user model nor a domain model: the prompts delivered are linked to each text a priori. *Writing Partner* (Zellermayer et al., 1991) is another system developed within the same framework: it delivers guidance during the writing process through diverse predefined metacognitive questions for assessing writing quality. Although these systems have been positively tested some drawbacks remain: (a) there are few differences between control group and experimental group, although they are attributed to the non-solicited prompts group, (b) there is no adaptivity to the student. Second generation of SRL systems aims at integrating SRL management into ITS with such adaptivity. We now present systems in this line of research according to the main processes involved: reading, writing or learning.

1. Reading-Centered SRL-systems

Aist (2000) developed *LISTEN*, a reading tutor equipped with speech recognition that listens to students and helps them learn to read. Text read aloud sentence per sentence by students is analyzed by *LISTEN*, which in turn provides online feedback: speaking a word or several words, spelling out aloud a word by phonemes or by syllables. Vocabulary help is available through WordNet. *LISTEN* also allows students to read stories close to both their interest and their reading ability. Although this system is mainly devoted to reading, another module allows students to write stories.

2. Writing-centered SRL-systems

Boylan and Vergaro (1999) designed the *Business Letter Tutor* (BLITS), a case-based system that helps secretaries to write business letters. First the system gets some characteristics of the letter to be written, then it delivers to the secretary the closest letter from its database. Each letter of the database is indexed with meta-tags and a neural network-based processing performs the matching.

3. Learning-centered SRL-systems

Graesser and his colleagues (Graesser, Person, & Harter, in press) are developing *Autotutor*, a LSA-based ITS in the domain of computer literacy. One of the purposes of *Autotutor*, for which LSA is required, is to answer unrestricted student questions by matching the latter with domain content. Another functionality of *Autotutor*, named "*Selection of next good answer aspect*" (GAA) uses LSA to select the next dialogue content to cover, within a set of pre-computed GAA. This selection is computed according to three criteria: (a) the matching of the next GAA with the zone of proximal development (*Autotutor* selects the next GAA to cover which similarity with the previous one would be both *below* a specified threshold—too similar, i.e., already covered—and still *above* other less related aspects); (b) its coherence (i.e., semantically close) with the preceding GAA; (c) its resemblance with the uncovered content.

4. Comments

Most of the systems presented above are both prompt-based and domain-dependent. First, they rely on a domain theory (i.e., composition process, misconceptions on a specific domain) to propose prompts that should influence students' metacognitive processes (i.e., "What more do you need to know in order to solve this problem?"). Zellermayer et al. (1991) showed that such prompts require a low cognitive load only if they are unsolicited. Boekaerts (1997) showed that direct teaching of such strategies did not systematically entail a long term use. We showed above that prompt delivering has some drawbacks: weak portability to another domain content, difficult prompt interpretation—prompts can either be problematic for novices or ignored by experts.

Secondly, they are domain-dependent. Current ITS generally contain a predefined knowledge structure that is used to select content. This structure is required to formulate a lesson goal given the student profile. This structure also both mimics expert domain knowledge and constraints further implementations to other domain contents.

Our system aims at two goals: no delivering metacognitive prompts and being domain-independent. It also includes some features of the previous systems. Firstly, we decided to offer a minimal guidance to the student in order to promote autonomy: the student will be responsible to stop at any time the delivery of content material to

enter the assessment part and vice versa. Secondly, in order to promote SRL processes, students are asked to judge their learning, just following the exposition to the content. Finally, it is both comprehension and production-centered, like *LISTEN*. In order to tightly link both processes, we decided to rely on a unique formalism to represent the content material and the student production. Both are represented as texts. However, the semantics of texts needs to be represented since the system has to perform comparisons between student production and content. The "next text to be read" like in *Autotutor* or *LISTEN* is also selected according to a semantic representation. For these reasons, all texts are represented in the LSA formalism, a model which is now described.

Presentation of the system

5. Latent Semantic Analysis: a knowledge representation system for Apex 2.0

As we mentioned earlier, our system is based on two main loops. The first one selects texts to be read by the student, the second one assesses learning by comparing a text written by the student to the texts he or she has read. Both processes require a semantic match between texts. Therefore, we relied on LSA, a tool which performs semantic comparison between texts based on the statistical analysis of huge corpora (Deerwester, Dumais, Furnas, Landauer, & Harshman, 1990; Landauer, Foltz, & Laham, 1998). LSA is based on the idea that two words are semantically similar if they statistically occur in semantically similar paragraphs all over the corpus. Two paragraphs are said semantically similar if they contain semantically similar words. For instance, *weapon* and *gun* are semantically similar since they appear statistically in similar paragraphs, that is, they occur with similar words (*shot*, *military*, *crime*, etc.). This mutual recursion is solved by a form of factorial analysis which assigns each word to a high-dimensional vector (about 300 dimensions).

Each word being represented by a vector, it is straightforward to assign each new sequence of words to a new vector, by simple sum of vectors. If we consider that each knowledge source can be represented by a sequence of words, we can say that LSA is a knowledge representation system. Of course, it is not efficient for reasoning on the representation, for instance drawing inferences, since the 300 values defining a vector do not have an intelligible meaning per se. Semantic networks or description logics are much more powerful formalisms for this purpose. However, LSA is more appropriate for comparing representations, which is not the case for symbolic formalisms. A semantic comparison of two sequences of words, i.e., two vectors, is performed by simply computing the cosine of their angle. A semantic comparison is therefore a number between –1 (no similarity) and 1 (highest similarity). The number of dimensions plays an important role. If the number of dimensions is too

small, too much information is lost. If it is too big, not enough dependencies are drawn between vectors. Dumais (1991) empirically showed that a size of 100 to 300 gives the best results in the domain of language. We will not present in detail the mathematical aspects of LSA which are detailed elsewhere (Deerwester et al., 1990).

We claim that LSA is an adequate tool for our purpose for several reasons. Firstly, Dumais (1991) showed that LSA improves text retrieval performance up to 30 %, compared with standard methods. Secondly, another line of research on LSA showed its capabilities to assess comprehension through text exposition (Foltz, 1996; Rehder, Schreiner, Wolfe, Laham, Landauer, & Kintsch, 1998; Wolfe et al., 1998). Actually, these authors showed that LSA is as good as humans to assess the comprehension of students who have read a set of texts on a given topic and produced an essay about this topic. Thirdly, a more recent line of research showed LSA's capabilities to model user within an ITS (Graesser et al., in press; Zampa & Lemaire, 2002). LSA can be used to find the most appropriate text to provide to a student, given the texts read. The idea, close to *Autotutor*'s GAA, is to select a text which is neither too close nor too far from the student model.

6. Previous versions of Apex

Our system is a follow-up of previous versions of *Apex* presented elsewhere (Dessus, Lemaire & Vernier, 2000; Lemaire & Dessus, 2001). We describe here the main functionalities of these versions in order to highlight what is new in *Apex 2.0*. *Apex* is a domain-independent ITS that can automatically assess the semantic content of student essays.

- *Apex 1.0* takes as input a student essay and returns an assessment based on a semantic comparison with the relevant parts of the course. For each relevant unit of the course, the system informs the student about the way the content is covered. For instance, the system could say "*You have covered very well the section about X. You have covered badly the section about Y*". Then the student is asked to work on the essay once more. Assessments at the paragraph level were delivered as well as a general grade. We compared this last measure with human grades and found a correlation of 0.59.
- *Apex 1.2* is a Web version of the system, written in PHP and MySQL. It can manage several students, several teachers and several courses. A student can pick up a course, select a topic, write an essay, get an assessment, revise the essay accordingly, etc. Teachers can add new courses and structure them.
- *Apex 1.5* analyses the student essay in order to split it into coherent parts. Instead of comparing the essay as a whole, each part is compared with the relevant parts of the course. Therefore, the indications to the student are more precise. Consequently, the correlation with human grades raises to 0.62.

These systems assess student essays but they do not help students to improve their knowledge. The student

would know which part of the course was not well understood but the student might not know what to do to fill this gap. In other words, these systems take care of the control but not of the learning. Therefore, our goal was to design a system that would do both. In addition, we thought that it would be important to take into account student comprehension. It is useful to know whether the student has learned something about the text that has been presented. This information could be used in the process of selecting the next text.

7. An overview of Apex 2.0

The Reading and Writing loop (see Figure 2 below) both rely on three kinds of marker associated to each text (i.e., provided, summarizable, deliverable). Each marker indicates:

? whether the text has been already *provided* in the current Reading loop. This is to ensure that the system is not providing the same text twice in the same Reading loop. Initially, all texts are marked as not provided;

? whether the text has been judged to be *summarizable* or not. This information is asked after the student has read a text. It is used for comparison with the system assessment. We could have directly asked the student to indicate whether the text was learned or not but we think this would be too hard a question. We think it is better to get this information by means of a question related to the way the text could be summarized, especially because this is the task the student will perform in the Writing loop. Initially, all texts are marked as not summarizable;

? whether the text is *deliverable* or not. Texts that are both judged as summarizable by the student and positively assessed by the system are permanently marked as non-deliverable. They will no longer be provided to the student. A text is positively assessed if it is semantically close enough to the text written by the student. Initially, all texts are deliverable.

All this information is represented into a special file updated after each student action.

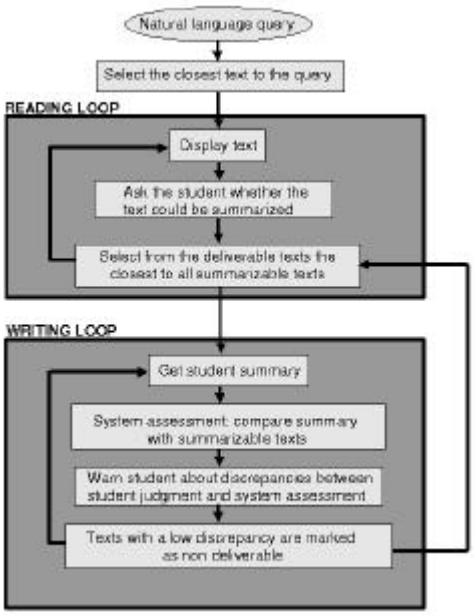

Figure 2 – Apex architecture

Using *Apex 2.0*, the student enters a Reading/Writing loop, each one in itself also being a loop.

The Reading loop. The goal of the Reading loop is to provide the student with texts. Texts are selected according to an initial natural language query formulated by the student and to what the system knows about the student from the beginning of the session. We tested the system with a corpus in sociology of education (56 documents, 29,024 words). We also added a 137,598-words corpus in the same domain which was only used for improving the semantic comparison accuracy. All these texts were beforehand analyzed by LSA and represented as vectors in a 300-dimensions semantic space. Suppose a student composes the following query*: "I'd like to work on the school effects"*. *Apex 2.0* would find the closest text to this query and provide the student with this text. The student reads the text. Then the student is required to click on one of the two buttons: "*I think I could sum it up*" or "*I don't think I could*" (cf. left screen of Fig. 3). If the first choice is selected, the text is marked in the system as "summarizable".

The system then selects the next text which is, from all deliverable texts not already provided in the current Reading loop, the closest text to all summarizable texts. The idea is to select a text which content is mostly related to what the student is supposed to have learned. To do that, the system computes the average semantic proximity between each text and all summarizable texts. If no text has been marked as summarizable yet, it means that the system is not able to select the next text. Therefore, the student is asked to rephrase the initial query. After a while, the student can decide that enough texts have been provided. The student then enters the Writing loop.

The SRL process solicited in this loop is concerned with cognitive monitoring. In confronting their reading goals to the texts proposed by the system, students can refine them. In so doing, students can get a more precise view of the content to learn.

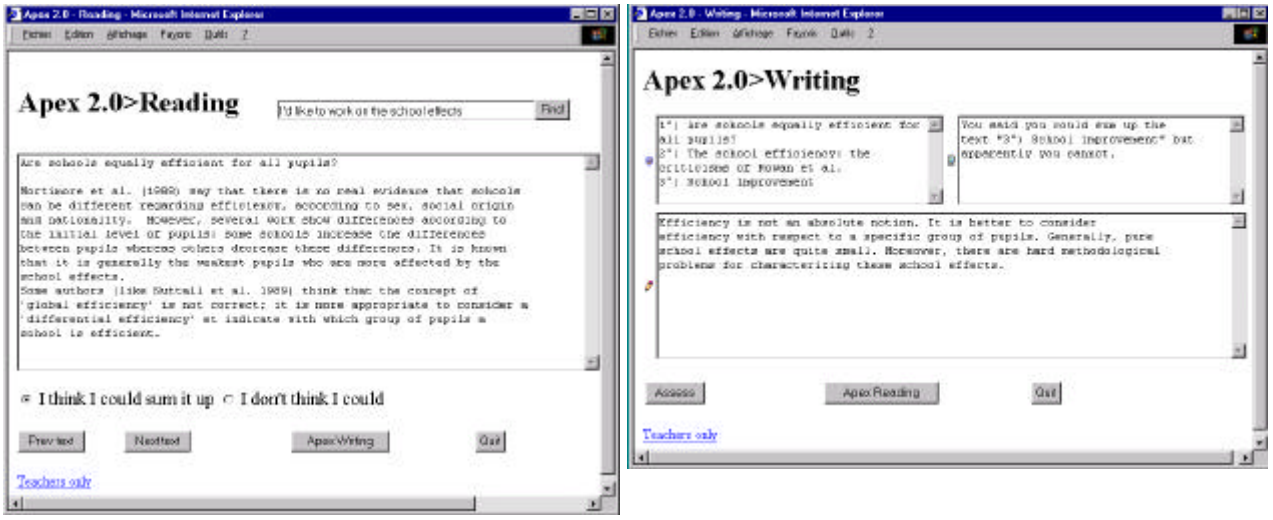

Figure 3 – Screendumps of *Apex 2.0*. Left screen: *Reading module*. Right screen: *Writing module*

The Writing loop. The student writes out a text summarizing the content of all texts read during the Reading loop. Of course, only the texts judged as summarizable are supposed to be summarized. The system displays the title of all of them in a separate pane (upper left pane of the right screen of Fig. 3). The goal of the Writing loop is to know whether the student is really able to summarize these texts. The way to do that is to compare two assessments: the first one, performed by the student after each text has been read (i.e., summarizable or not) and the second one performed by LSA (i.e., deliverable or not). This latter assessment is performed in the following way: the student summary is computed by LSA and represented by a 300-dimensions vector. This vector is then compared with all the texts the student has previously judged as summarizable. If the similarity is high enough (a threshold is used which is arbitrarily set at 0.5), the text is marked as non deliverable. If the similarity is too low the text remains deliverable and the student is warned: "*You said you could summarize the text X but apparently you cannot*". This information can be used by students to revise their summary. In that case, the Writing loop starts again. Otherwise, the student re-enters the Reading loop.

The SRL process dedicated to this loop is cognitive control. In writing a summary that is semantically compared with the source texts, the student can keep in mind the parts of these texts that are not summarized well. In turn, the summary can be reviewed to more adequately match with the source texts. This way the student is able to have a better grasp of the content delivered during this second loop.

Further work and limits

We plan to test *Apex 2.0* in experimental conditions, in which subjects perform various learning goals (e.g., reading for learning, studying for an exam, completing a brief essay). We will measure some dependent variables such as: number of goal refinements, number of texts read, texts estimated as learned, number of revisions of the text, and so on. A next version of *Apex* will address some limits of our system. Among others, we plan to add the following functionalities: (a) the judgment of learning could be refined by allowing the student to highlight parts of texts that were considered as learned; (b) text deliverability could also be a priori defined by the teacher, in order to allow content selection by the teacher; (c) in order to help students who begin learning a content, teacher predefined queries would be proposed to them; (d) in order to scaffold the student activity it would be adequate to diminish progressively the help provided.

Discussion

We argued above that *Apex 2.0* can be distinguished from other SRL-based systems on two points:

(1) Most ITS use separate knowledge bases to model respectively users, pedagogical moves, and content. The knowledge is formalized either by the teacher or the domain expert. Our system imposes neither a specific formalization nor a rigid model because all the necessary knowledge resides within a unique formalism.

(2) Current systems are prompt-based, i.e., systems usually provide metacognitive hints that are (or not) taken into account by users. Our system is rather an attempt to emphasize the gap between the content judged to be learned and the content actually learned. In that way students can try to reduce this gap either by writing better summaries or by judging more objectively their learning.

Two questions remain to be investigated. (1) The question of transfer: adequate metacognitive prompts could lead to internalization by the user, who would be able to perform metacognitive skills *without* guidance. To date, this domain is seldom inquired, and we would test if abilities developed within *Apex 2.0* remain after its use. (2) Although our system is design to promote SRL—and consequently learning—through reading and writing, how human beings learn that way is not yet elicited. Scardamalia and Bereiter (1991) proposed a framework in which expertise in reading and writing is viewed as dialectical processes that contribute to increase a domain expertise. We will use this framework in further research, in which we will test *Apex 2.0* in context.

Acknowledgements

We thank Erica de Vries and Beulah Camou for their thoughtful comments on an earlier version of this paper, and Pascal Bressoux for providing the text of the course.

References

Aist, G. (2000). Human Tutor and Computer Tutor Story Choice in Listening to Children Read Aloud. *Communication at the Workshop of ITS-2000 "Modeling human teaching tactics and strategies"*. Montreal.

Azevedo, R. (2001). Using hypermedia to learn about complex systems: A Self-Regulation Model. *Communication at the Help Workshop of AI-ED 2001 Conference.* San Antonio.

Boekaerts, M. (1997). Self-regulated learning: a new concept embraced by researchers, policy makers, educators, teachers, and students. *Learning and Instruction, 7*, 161-186.

Boylan, P., & Vergaro, C. (1999). Metacognition in epistolary rhetoric: A case-based system for writing effective business letters in a foreign language. In S. P. Lajoie & M. Vivet (Eds.), *Artificial Intelligence in Education* (pp. 305-312). Amsterdam: IOS Press.

Butler, D. L., & Winne, P. H. (1995). Feedback and self-regulated learning: A theoretical synthesis. *Review of Educational Research, 65*, 245-281.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science, 41*, 391-407.

Dessus, P., Lemaire, B., & Vernier, A. (2000). Free-text assessment in a Virtual Campus. In K. Zreik (Ed.), *Proc. International Conference on Human System Learning (CAPS'3)* (pp. 61-76). Paris: Europia.

Dumais, S. T. (1991). Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments and Computers, 23*, 229-236.

Foltz, P. W. (1996). Latent semantic analysis for text-based research. *Behavior Research Methods, Instruments and Computers, 28*, 197-202.

Graesser, A. C., Person, N. K., & Harter, D. (in press). Teaching tactics and dialog in Autotutor. *International Journal of Artificial Intelligence in Education.*

Landauer, T. K., Foltz, P. W., & Laham, D. (1998). An introduction to Latent Semantic Analysis. *Discourse Processes, 25*, 259-284.

Lemaire, B., & Dessus, P. (2001). A system to assess the Semantic Content of Student Essays. *Journal of Educational Computing Research, 24*, 305-320.

Lesgold, A. (1988). Toward a theory of curriculum for use in designing Intelligent Tutoring Systems. In H. Mandl & A. Lesgold (Eds.), *Learning Issues for Intelligent Tutoring Systems* (pp. 114-137). Berlin: Springer.

Nelson, T. O. (1999). Cognition versus Metacognition. In R. J. Sternberg (Ed.), *The Nature of Cognition* (pp. 625-644). Cambridge: MIT Press.

Rehder, B., Schreiner, M. E., Wolfe, M. B. W., Laham, D., Landauer, T. K., & Kintsch, W. (1998). Using Latent Semantic Analysis to assess knowledge: some technical considerations. *Discourse Processes, 25*, 337-354.

Salomon, G., Globerson, T., & Guterman, E. (1989). The computer as a Zone of Proximal Development: Internalizing Reading-related metacognitions from a Reading Partner. *Journal of Educational Psychology, 81*, 620-627.

Scardamalia, M., & Bereiter, C. (1991). Literate expertise. In K. A. Ericsson & J. Smith (Eds.), *Toward a general theory of expertise* (pp. 172-194). Cambridge: Cambridge University Press.

Self, J. (1994). The role of Student Models in Learning Environments. *Transactions of the Institute of Electronics, Information and Communication Engineers, E77-D*, 3-8.

Veermans, K., de Jong, T., & van Joolingen, W. R. (2000). Promoting self-directed learning in simulation-based discovery learning environments through intelligent support. *Interactive Learning Environments, 8*, 229-255.

Wolfe, M. B. W., Schreiner, M. E., Rehder, B., Laham, D., Foltz, P., Kintsch, W., & Landauer, T. K. (1998). Learning from text: Matching readers and texts by Latent Semantic Analysis. *Discourse Processes, 25*, 309-336.

Zampa, V., & Lemaire, B. (2002). Latent Semantic Analysis for User Modeling. *Journal of Intelligent Information Systems, 18*, 15-30.

Zellermayer, M., Salomon, G., Globerson, T., & Givon, H. (1991). Enhancing Writing-Related Metacognitions through a Computerized Writing Partner. *American Educational Research Journal, 28*, 373-391.