

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

Philippe Dessus and Benoît Lemaire

Laboratoire des Sciences de l'Éducation  
Université Pierre-Mendès-France  
BP 47 – 38040 Grenoble Cedex 9, France

{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 *Apex*, a two-loop system which provides texts to be learned through summarization. In the first loop, called *Reading*, 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. In the second loop, called *Writing*, 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.

## 1 Introduction

This paper is about Lesgold's "two fundamental laws of instruction" [1]: (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.

## 2 *Apex* = ILE + SRL

To date, systems called ILE are either *production-centered*, qualitatively assessing production to help students write texts [2,3] or solve problems [4]; or *comprehension-centered* [5,6], providing students with texts related to their comprehension of previously delivered texts. The main features of ILEs are the following [7]: (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, etc.), (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, etc.).

We combine this notion with another line of research: SRL, which relies on four assumptions [8]: (a) the learner is an active 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 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 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*.

## 3 Self-Regulated Learning

### 3.1 Theoretical Framework

We adapted the model of Self-Regulated Learning, detailed in Butler and Winne [9]. These authors describe their model as follows (see Fig. 1): firstly, 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 *controlled* [10]. In other words, learning occurs when the teacher helps students to formulate and monitor their own goals 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.

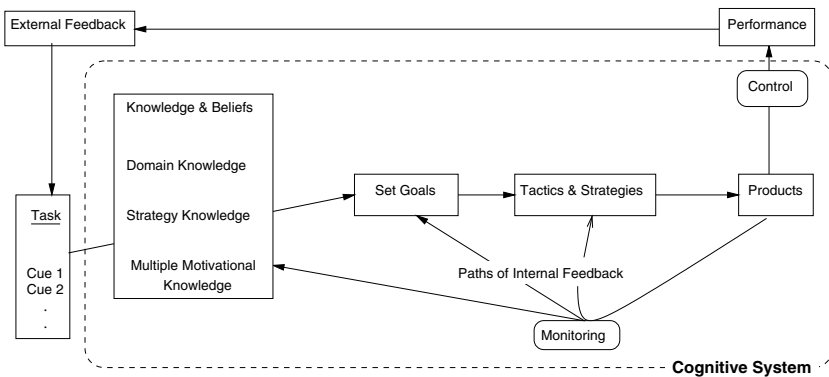


Fig. 1. A Model of Self-Regulated Learning (after [9,10])

### 3.2 Related Work on Computer-Aided SRL

One of the first SRL-centered computer programs was *Reading Partner* [6]. 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* [3] 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.

**Reading-Centered SRL-systems.** *Listen* [5] is 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 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.

**Writing-centered SRL-systems.** *Business Letter Tutor* [2] is 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.

**Learning-centered SRL-systems.** *Autotutor* [11] is 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 them 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.

**Comments.** Most of the systems presented above are both prompt-based and domain-dependent. Firstly, 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. [3] showed that such prompts require a low cognitive load only if they are unsolicited. Boekaerts [12] 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, current ITS are domain-dependent. They generally contain a pre-defined 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, our system both comprehension and production-centered, like *Listen*. In order to tightly link both processes, we 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.

## 4 Presentation of the System

### 4.1 LSA: 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 matching between texts. Therefore, we relied on LSA, a tool which performs a semantic comparison between texts based on the statistical analysis of huge corpora [13,14]. Two words are considered 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 (i.e., they occur with similar words like *shot*, *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 set of words to a new vector, by simple sum of vectors. If we consider that each knowledge source can be represented by a set 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 sets 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 [15] 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 [13].

We claim that LSA is an adequate tool for our purpose for several reasons. Firstly, Dumais [15] 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 [16]. 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 [17]. Thirdly, a more recent line of research showed LSA's capabilities to model user within an ITS [11,18]. 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.

## 4.2 Previous Versions of *Apex*

Our system is a follow-up of previous versions of *Apex* presented elsewhere [19]. 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 .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, indications to the student are more precise. Consequently, the correlation with human grades raises to .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 to select the next text.

## 4.3 An Overview of *Apex 2.0*

The Reading and Writing loops (Fig. 2) rely on three kinds of markers associated to each text: *provided*, *summarizable*, *deliverable*. This information is updated after each student action. Each marker indicates:

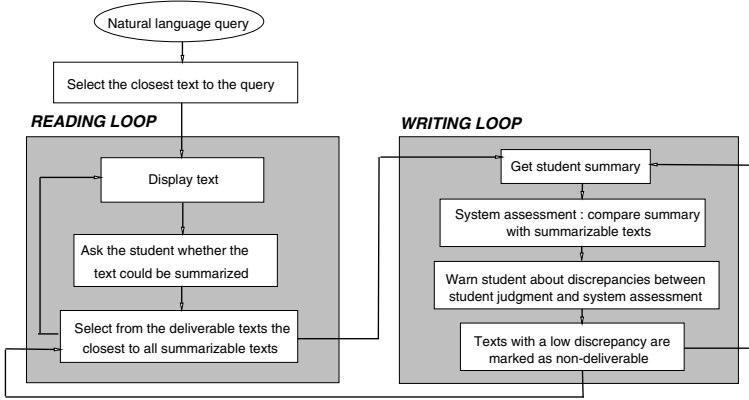


Fig. 2. Apex architecture

- 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.

**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*”. If the first choice is selected, the text is marked in the system as summarizable.

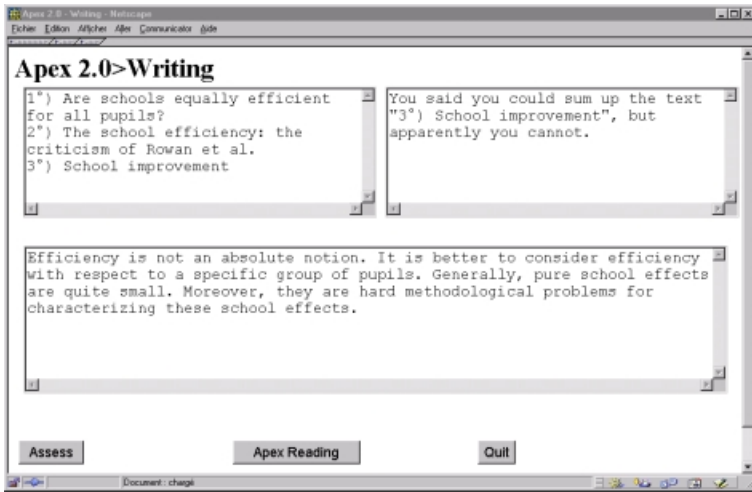


Fig. 3. Screenshot of *Apex 2.0* Writing module

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 here is concerned with cognitive monitoring. In confronting their reading goals to the texts proposed by the system, students can refine them. They can thus get a more precise view of the content to learn.

**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 (upper left pane 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 as follows: the student summary is computed by LSA and represented by a vector. This vector is then compared with all the texts the student has previously judged as summarizable. If the similarity is high enough (above a threshold arbitrarily set at .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 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 loop.

## 5 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, number of revisions of the text, etc. A next version of *Apex* will address some current limits: (a) the judgment of learning could be refined by allowing the student to highlight parts of texts 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.

## 6 Discussion

*Apex 2.0* can be distinguished from other SRL-based systems on two points: a unique formalism and no intrusive hints. Firstly, most ITS use separate knowledge bases to model respectively users, pedagogical moves, and content. 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. Secondly, 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.

The question of transfer remains to be investigated. 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.

**Acknowledgements.** We thank E. de Vries and B. Camou for their thoughtful comments on an earlier version of this paper, and P. Bressoux for providing the text of the course.

## References

1. Lesgold, A.: Toward a theory of curriculum for use in designing intelligent tutoring systems. In Mandl, H., Lesgold, A., eds.: *Learning Issues for Intelligent Tutoring Systems*. Springer, Berlin (1988) 114–137
2. Boylan, P., Vergaro, C.: Metacognition in epistolary rhetoric: A case-based system for writing effective business letters in a foreign language. In Lajoie, S.P., Vivet, M., eds.: *Artificial Intelligence in Education*. IOS Press, Amsterdam (1999) 305–312
3. Zeller Mayer, M., Salomon, G., Globerson, T., Givon, H.: Enhancing writing-related metacognitions through a computerized writing partner. *American Educational Research Journal* **28** (1991) 373–391
4. Veermans, K., de Jong, T., van Joolingen, W.R.: Promoting self-directed learning in simulation-based discovery learning environments through intelligent support. *Interactive Learning Environments* **8** (2000) 229–255
5. Mostow, J., Aist, G.: Evaluating tutors that listen. In Forbus, K.D., Feltovich, P.J., eds.: *Smart Machines in Education*. AAAI, Menlo Park (2001) 169–234
6. Salomon, G., Globerson, T., Guterman, E.: The computer as a zone of proximal development: Internalizing reading-related metacognitions from a reading partner. *Journal of Educational Psychology* **81** (1989) 620–627
7. Self, J.: The role of student models in learning environments. *Trans. of the Institute of Electronics, Information and Communication Engineers* **E77-D** (1994) 3–8
8. Azevedo, R.: Using hypermedia to learn about complex systems: A self-regulation model. In: *Help Workshop of the AI-ED 2001 Conference*, San Antonio (2001)
9. Butler, D.L., Winne, P.H.: Feedback and self-regulated learning: A theoretical synthesis. *Review of Educational Research* **65** (1995) 245–281
10. Nelson, T.O.: Cognition versus metacognition. In Sternberg, R.J., ed.: *The Nature of Cognition*. MIT Press, Cambridge (1999) 625–644
11. Graesser, C., Person, N., Harter, D., the Tutoring Research Group: Teaching tactics and dialog in Autotutor. *International Journal of Artificial Intelligence in Education* **12** (in press)
12. Boekaerts, M.: Self-regulated learning : a new concept embraced by researchers, policy makers, educators, teachers, and students. *Learning and Instruction* **7** (1997) 161–186
13. Deerwester, S., Dumais, S., Furnas, G., Landauer, T., Harshman, R.: Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science* **41** (1990) 391–407
14. Landauer, T., Foltz, P., Laham, D.: An introduction to Latent Semantic Analysis. *Discourse Processes* **25** (1998) 259–284
15. Dumais, S.: Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments and Computers* **23** (1991) 229–236
16. Wolfe, M., Schreiner, M., Rehder, B., Laham, D., Foltz, P., Kintsch, W., Landauer, T.: Learning from text: Matching readers and texts by Latent Semantic Analysis. *Discourse Processes* **25** (1998) 309–336
17. Kintsch, E., Steinhart, D., Stahl, G., Matthews, C., Lamb, R., the LSA Group: Developing summarization skills through the use of LSA-based feedback. *Interactive Learning Environments* **8** (2000) 87–109
18. Zampa, V., Lemaire, B.: Latent Semantic Analysis for user modeling. *Journal of Intelligent Information Systems* **18** (2002) 15–30
19. Lemaire, B., Dessus, P.: A system to assess the semantic content of student essays. *Journal of Educational Computing Research* **24** (2001) 305–320