Latent Semantic Analysis for User Modeling

Virginie Zampa and Benoît Lemaire
L.S.E.
University of Grenoble II
BP 47
38040 Grenoble Cedex 9
France

Abstract. Latent Semantic Analysis (LSA) is a tool for extracting semantic information from texts as well as a model of language learning based on the exposure to texts. We rely on LSA to represent the student model in a tutoring system. Domain examples and student productions are represented in a high-dimensional semantic space, automatically built from a statistical analysis of the co-occurrences of their lexemes. We also designed tutoring strategies to automatically detect lexeme misunderstandings and to select among the various examples of a domain the one which is best to expose the student to. Two systems are presented: the first one successively presents texts to be read by the student, selecting the next one according to the comprehension of the prior ones by the student. The second plays kalah with the student in such a way that the next configuration of the board is supposed to be the most appropriate with respect to the semantic structure of the domain and the previous student's moves.

Keywords: Latent Semantic Analysis, User Modeling, Tutoring systems, Language Learning

1. Introduction

This paper describes a way to represent the student knowledge in a tutoring system by means of Latent Semantic Analysis (LSA). LSA is both a tool for representing the meaning of words (Deerwester et al., 1990) and a cognitive model of learning (Landauer and Dumais, 1997). LSA analyses large amount of texts by means of a statistical method and represents the meaning of each word as a vector in a high-dimensional space. Pieces of texts are also represented in this semantic space. Semantic comparisons between words or pieces of texts are made by computing the cosine between them. We rely on this tool to represent both domain and student knowledge in a tutoring system. In our field (language learning), domain knowledge is composed of the usual meaning of words as well as textual materials. Student knowledge is composed of the student meaning of words.

We designed two tutoring strategies based on this dual knowledge representation. The first one automatically detects student misunderstandings from the analysis of what he/she has written. The second

one selects among the various textual stimuli a student can be exposed to, the one which is supposed to be the best for improving learning.

We also extended the scope of LSA to cover other domains than language. Indeed, some domains can be described by means of sequences of lexemes, just like texts are composed of sequences of words. LSA is then used to compute the semantic proximities between these sequences of lexemes. Choosing the best sequence to expose the student to is also based on the representation of student knowledge in the semantic space. We apply these ideas in the domain of strategy game learning. Texts and words are just replaced by boards and pieces. We designed a system which helps a user learn the game by playing with this user and choosing the next move so that the new state of the game is optimal for learning.

2. Latent Semantic Analysis

LSA was primarily designed as a tool for text retrieval about ten years ago (Deerwester et al., 1990) but because of its favorable performance, its scope was extended to information filtering (Foltz and Dumais, 1992), cross-language information retrieval (Dumais et al., 1997) automatic grading of essays (Foltz, 1996; Lemaire and Dessus, to appear), measuring of text coherence (Foltz, 1998), assessment of knowledge (Rehder, 1998), machine learning (Lemaire, 1998) and then modelling human learning (Landauer and Dumais, 1997).

Before presenting LSA as a model knowledge representation, we will describe its principles.

2.1. LSA: a Tool for a Semantic Comparison of Texts

One of the problems in the field of text retrieval is to be able to retrieve pieces of texts given a list of keywords. However, because of polysemy, synonymy and inflexion, retrieving only the texts that contain one or more of the keywords does not work well. For instance, Steinbeck's book *Of Mice and Men* should be retrieved given the keywords *mouse* and *man* although none of these words appear in this form in the title. Therefore, retrieval should also be based on semantic information.

In order to perform such semantic matching, LSA relies on large corpora of texts to build a semantic high-dimensional space containing all words and texts, by means of a statistical analysis. This semantic space is built by considering the number of occurrences of each word in each piece of text (basically paragraphs). For instance, with 300 paragraphs and a total of 2,000 words, we get a 300x2,000 matrix. Each word is

then represented by a 300-dimensional vector and each paragraph by a 2,000-dimensional vector. Nothing new so far since it is just occurrence processing. The power of LSA lies in the reduction of these dimensions. It is this process that induces semantic similarities between words. All vectors are reduced by a method close to eigenvector decomposition to, for instance, 100 dimensions. The matrix X is decomposed as a unique product of three matrices: $X = T_0 S_0 D'_O$ such that T_0 and D_0 have orthonormal columns and S_0 is diagonal. This is called singular value decomposition. Then only the 100 columns of T_0 and D_0 corresponding to the 100 largest values of S_0 are kept, to obtain T, S and D. The reduced matrix $\overline{X}$ such that: $\overline{X} = TSD'$ permits all words and pieces of texts to be represented as 100-dimensional vectors. It is this reduction which is the heart of the method because it extracts semantic relations: if a word (e.g. bike) statistically co-occurs with words (e.g. handlebars, pedal, ride) that statistically co-occur with a second word (e.g. bicycle) *and* the first word statistically does not co-occur with words (e.g. flower, sleep) that do not co-occur with the second one, then the two words are considered quite similar. 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. A size of 100 to 300 gives the best results in the domain of language (Landauer and Dumais, 1997).

Similarities between words or pieces of texts are computed by means of the cosine between the corresponding vectors. The measure of semantic similarity is therefore a number between -1 and 1.

This method is quite robust: a word could be considered semantically close to another one although they never co-occur in texts. In the same way, two documents could be considered similar although they share no words. An interesting feature of the method is that the semantic information is derived only from the lexical level. There is no need to represent a domain theory by means of a semantic network or logic formulas.

Several experiments were performed which showed that LSA works quite well. One such experiment (Landauer and Dumais, 1997) consisted in building a general semantic space from a large corpora of English texts, then testing it with the synonymy part of the TOEFL test (Test Of English as a Foreign Language), which is composed of 80 questions. Given a word, the problem is to identify among 4 other words the one that is the semantically closest. LSA performed the test by choosing the word with the highest similarity between its vector and the vector of the given word. LSA results (51.5) compare with the average score (51.6) of foreign students admitted to American universities. To our knowledge, this is the first system able to pass such a standard test with no extra semantic knowledge added.

In several other studies (Foltz, 1996; Kintsch, to appear; Lemaire and Dessus, to appear; Wiemer-Hastings, 1999a; Wolfe, 1998) subjects were asked to write essays from a number of texts in a given domain. These essays were then ranked by human judges. Their task was to judge the adequation between the essay and the texts. In parallel, LSA was trained with the texts and ranked the essays according to the semantic proximity between each of them and the texts. LSA results compare again with the human results. Correlations between human judges and LSA are around 0.6 in all of these studies, which is similar to the usual correlation between different human judges grading the same text.

2.2. LSA: a Model of Learning

Apart from interesting results in the field of text retrieval and text comparison, LSA is also viewed as a model of language learning in the field of cognitive psychology. Psychological models might not be necessary for the design of tutoring systems. However, since our goal is to design tutoring strategies based on a representation of the student knowledge, we found interesting to base our representation on a psychologically valid representation.

First of all, we will describe this model but we will see later that we can extend it to other kinds of knowledge.

LSA learns the meaning of words in the same way a child does: by reading texts[1]. An interesting question is whether the two rates of learning are similar. This model has been tested by simulating word learning between age 2 and 20 (Landauer and Dumais, 1997). The authors estimated that human beings read about 3500 words a day, and learn an average of 7 to 15 words per day during that period. If provided with a similar amount of texts, LSA learns 10 words a day to get a performance similar to the humans by the age of 20 (defined as the performance to the TOEFL test described earlier). This result is coherent with the human rate of learning. A similar method (named HAL) based on a high-dimensional representation shows its ability to model the human semantic memory (Lund, 1996; Burgess and Lund, 1997). Therefore, LSA seems an adequate way of representing semantic knowledge.

We extended that model in the following way:

– A domain D is composed of lexemes. In the domain of language, lexemes are words. In the domain of problem solving, lexemes

[1] Most of the words we know, we learn from reading since (1) spoken language contains just a small part of the language; (2) very little vocabulary is learned from direct instruction (Landauer and Dumais, 1997).

are facts and conclusions (`high-fever`, `prescribe-penicillin`, `meningitis`, etc. in the domain of medicine). In the domain of game playing, lexemes are positions of pieces (`pawn-in-E2`, `queen-in-F4`, etc. in chess).

- A student learns the domain by being exposed to sequences of lexemes (sequences of words, sequences of facts and conclusions, sequences of pieces' positions, etc.).

- What is learned is semantic similarities between lexemes or sequences of lexemes (for instance, `high-fever` and `meningitis`). In chess, two boards can be semantically similar although their pieces are not in the same positions; it is a characteristic of chess masters to be able to recognize two boards as being similar (see the famous de Groot's experiments).

- LSA predicts the semantic similarity between two lexemes or sequences of lexemes given the sequences of lexemes the student has been exposed to.

Some of the sequences of lexemes that are presented to the student will highly improve learning because their structure map the semantic structure of the domain. Other sequences will be of poor interest for the student because they are either too close to or too far from the student knowledge. For instance, if 10 year-old children are provided with texts made for 6 year-old children, they will probably not learn much. In the same way, they will not learn much if given a text from Freud. The problem is therefore to find the optimal stimuli to expose the student to in order to maximize learning. Let us take another example from another domain. Exposure to chess boards generated by beginners will not allow the acquisition of the lexemes `castling`[2] or `fianchetto`[3]. In the same way, playing with masters might not be optimal because the underlying strategy could be too subtle to grasp.

Knowing which sequence is best for the student depends on:

- the semantic structure of the domain;

- the student knowledge.

These two kinds of knowledge are also necessary for automatically detecting student misunderstandings of lexemes. If both are represented in the same formalism, it is possible to detect these misunderstandings

[2] Special move which consists in moving the king two squares towards the rook and moving the rook to stand to the opposite side of the king.

[3] Special position of the bishop, the king and 3 pawns.

by measuring the difference between the student meaning of a lexeme and the "right" meaning of the same lexeme (for instance, detecting that the meaning of `eclipse` for the student, defined from the LSA analysis of texts written or pronounced by this student, is far from the meaning of `eclipse` defined from a LSA analysis of a textbook in astronomy). We will present some mechanisms that deal with that problem in a following section.

The previous discussion justifies the need to represent domain knowledge and student knowledge in the same LSA formalism of knowledge representation. Subsequently, we will be able to design tutoring strategies for selecting the sequences of lexemes that are best to present to a given student or to detect lexeme misunderstandings. All this is akin to the well-known structure of tutoring systems (Wenger, 1987): expert module, user model and pedagogical module.

3. High-dimensional Representation of Knowledge

First, we will present the way LSA can represent domain knowledge, then, how the same formalism can be used to represent student knowledge.

3.1. High-Dimensional Representation of Domain Knowledge

One of the main interests of our approach is that the representation of domain knowledge is automatically built from examples. These examples should be semantically valid and therefore provided by experts (verbally or from books). For instance, in the domain of language learning, examples are just well-formed texts. In the domain of game playing, examples are boards as well as an indication whether it is a winning board or a losing board (this information is obtained at the end of the game).

From these examples, LSA builds a semantic space in which all lexemes and examples are represented. There is no need to hand-build semantic networks or logic formulas to represent the domain. All is automatically done by LSA. The only requirement is to find an adequate formalism to represent the examples.

Let us take an example in a domain other than language. For instance, in the game of tic-tac-toe, lexemes could be moves. Tic-tac-toe is played on a 3x3 board. Each of the 9 squares might contain `x`, `o` or a blank. Therefore, the tic-tac-toe "language" contains 27 lexemes: `x1`, `o1`, `b1`, `x2`, `o2`, `b2` ... `x9`, `o9`, `b9` (Squares are labelled from 1 to

9, starting upper left). An example in this domain is a sequence of lexemes, that is a board..

For instance the following board is represented by the sequence `x8 o5 x7 o9 x6 o1 losing-for-x`:

o		
	o	x
x	x	o

We need to represent all possible stimuli. Therefore, non-final boards are also represented as well as an indication of the final score for this particular game. For instance: `x5 o2 winning-for-x`. Other games beginning with the same configuration might lead to a loss for x: we might have several `x5 o2 losing-for-x` in the semantic space. However, from a statistical point of view, the most probable result will be the closest to `x5 o2`[4].

All examples are processed by LSA as if they were texts: examples (sequences of lexemes) are analogous to texts and components of examples (lexemes) are analogous to words. The high-dimensional semantic space therefore contains all lexemes and all examples. For instance, if enough examples are provided, the lexeme `x5` (central square) will be closer to the winning boards than to the losing boards, although the well-known information that playing the central square is good was never represented as such.

To make sure that this knowledge representation is relevant and more efficient than a non-semantic representation, we performed the following experiment. We compared the learning rate of a program playing tic-tac-toe based on a LSA representation of previous games and the same program without LSA representation. These two programs look for the previous board which is most similar to the current board. If it was a winning game, they play like previously, otherwise they play the opposite (see (Lemaire, 1998) for more details). The first program uses LSA to look for the closest previous board whereas the second one selects the prior board that has the most elements in common with the current board.

Both programs were run on 1400 games. Figure 1 shows the learning curves. The x-axis is the number of games and the y-axis is the cumulative number of games won. As expected, the LSA-based algorithm plays very badly at the beginning. However, after 250 games, the learning curve of the LSA-based algorithm becomes higher. A χ^2 test shows

[4] It is worth noting that scores can be more elaborate than only `winning` or `losing`. To do so, more lexemes can be defined as `highly winning`, etc. Numerical lexemes can also be used.

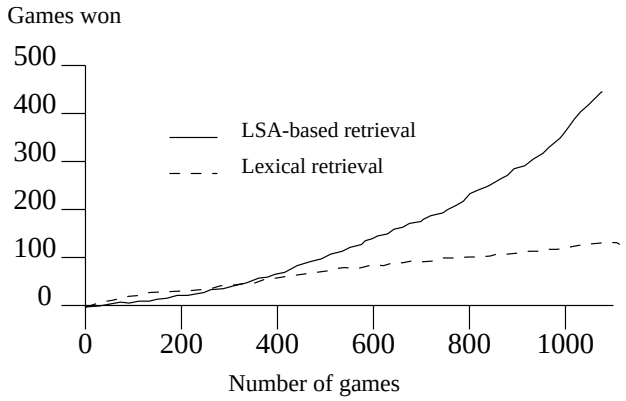

Figure 1. Results for the tic-tac-toe problem

that the different is significant ($p < .0001$). This test provides another evidence for the representation of knowledge based on LSA.

It is worth noting that we do not need to represent all possible examples of the domain, but only enough of them to statistically capture a significant part of the latent semantic structure of the domain.

3.2. High-Dimensional Representation of Student Knowledge

In the same way, we represent the student knowledge, that is the student's meaning of entities (we call entity an element of the semantic space, either a lexeme or a sequence of lexemes). Lexemes are therefore described twice:

- as a domain entity to represent the standard meaning of the term, constructed as shown previously from the word usage in the language;
- as a student entity to represent the student's meaning of the term.

For instance, the semantic space may contain one instance of `pneumonia` as a domain entity and one instance as a student entity. In the same way, sequences of lexemes are represented in the semantic space. Before being represented, that knowledge needs to be extracted. There are several ways to do that:

- student productions are recorded: student entities are therefore different from domain entities;
- all the sequences of lexemes the student was exposed to and tested in are represented in the space. That way, student entities are

domain entities that can be weighted by a score corresponding to the comprehension of the domain entity by the student.

Student productions can be written or spoken productions. In the first case, texts written by the student are analysed by LSA. The problem is that we need enough texts to be able to have a workable representation. Another solution, that we do not explore, would be to rely on speech production. By recording a user discourse and using a speech recognition system, it would be possible to get a large corpus. In that case, knowledge representation would be very accurate. For instance, it should be possible to record everything said by a child over a week or so, by means of a portable microphone and a cordless connection to a computer. We will see in the next section that such a device would allow the automatic detection of word misunderstandings.

In the case of learning, the overall goal is that the student entities cover all the domain entities. If the student entities are all in the same part of the space, it probably means that the student has a gap in his/her knowledge. In that case, the goal of the system would be to provide him with appropriate sequences of lexemes so that his/her knowledge covers a larger part of the space.

Now that we have a common representation of both domain and student knowledge, we need to design tutoring strategies. As we mentionned before, we designed two strategies: automatic selection of stimuli and automatic detection of misunderstandings.

4. Automatic Detection of Misunderstandings

As we have already seen, the meaning of a lexeme is given by all the lexemes close to it. This is akin to the Saussurian point of view that *the meaning (of a word) is determined by what surrounds it* (Saussure, 1993).

For instance, a LSA analysis of the "General reading up to 1st year college" database (available at `htpp://lsa.colorado.edu`) returns the following closest words to `pillow`: `bed` (0.81), `asleep` (0.71), `wake` (0.69), `awake` (0.69), `pillows` (0.68), `bedroom` (0.67), etc., which define the meaning of `pillow`.

Another example results from an analysis of a small database of animal features. The closest lexemes to the lexeme `eats meat` are `fawn-coloured` (.51), `tiger` (.36), `has black spots` (.20), etc.

That representation allows us to design a method to automatically detect lexeme misunderstandings. The idea is to take, for each lexeme written by te student, the neighbouring lexemes in the student semantic space. Then, we compare these semantic proximities in the

student semantic space and in the domain semantic space. If there is a too big difference between the semantic proximities in the two semantics spaces, it means that there is a student misunderstanding of that lexeme. For instance, if in the student space the word `pillow` is close to `drugs`, `codeine`, `aspirin`, `dosage`, we say that the student does not have a correct understanding of the word `pillow`.

To be more formal, for each lexeme X of the learner space, we consider the $X_1, \ldots, X_\alpha$ closest lexemes. Then, for each X_i, we compute the difference :

$$|proximity_{domainspace}(X, X_i) - proximity_{studentspace}(X, X_i)|$$

Therefore, we obtain α differences. The smaller these differences are, the better the understanding of X by the learner.

These differences are classified in two intervals : $[0; \lambda_1[, [\lambda_1; 2]$. The value λ_1 need to be defined experimentally. From our experience, we think that values such as 0.2 would be good starting points.

The understanding of X by the learner is determined from the distribution of the differences X_i over the two intervals:

- if most of the X_i belongs to $[0; \lambda_1[$, we consider that the meaning of X is well understood;
- else, we consider that the meaning of X is not well understood.

Figure 2 presents the algorithm. We implemented the previous "most of" by the fact that two third of the X_i belong to the corresponding category.

The list of lexemes that are likely to be misunderstood can then be used directly by a teacher or by the pedagogical module of a tutoring systems in order to select the appropriate learning materials.

5. Automatic Selection of Stimuli

As we mentioned before, in the LSA model, learning results from the exposure to sequences of lexemes. The idea that learning a second language is essentially based on the exposure to the language, and not only to explanation of the rules of that language, is nowadays recognized by researchers in second language acquisition (Krashen, 1988).

By being exposed to sequences of lexemes in a random fashion, a student would certainly learn some lexemes in the same way a child learns new words by reading various books. However, the process of learning could be speeded up by selecting the right sequence of lexemes given the current state of student entities. Therefore, the problem is

procedure misunderstood(X)
nb1=0
For each $X_1, \ldots, X_\alpha$ close to X
difprox=$|proximity_{domainspace}(X, X_i) - proximity_{learnerspace}(X, X_i)|$
if difprox $\in [0; \lambda_1[$: nb1 = nb1 + 1
end if
end for
if nb1 $> \frac{2}{3}\alpha$: write("X is probably well understood.")
else : write("X is probably not well understood.")
end if

Figure 2. Algorithm for the automatic detection of the misunderstanding of lexeme X

to know which text (for language learning) or which move (for game learning) has the highest chance of enlarging the part of the semantic space covered by the student entities.

5.1. Selecting the closest sequence

Suppose we decide to select the sequence which is the closest to the student sequences. Suppose that $\{s_1, s_2, \ldots s_n\}$ are the student sequences and $\{d_1, d_2, \ldots d_p\}$ the domain sequences, we select d_j such that:

$$\sum_{i=1}^{n} proximity(s_i, d_j)$$

is minimal. Figure 3 shows that selection in a 2-dimensional representation (remind that LSA works because it uses a lot of dimensions). Domain entities are represented by black squares and student entities by white squares

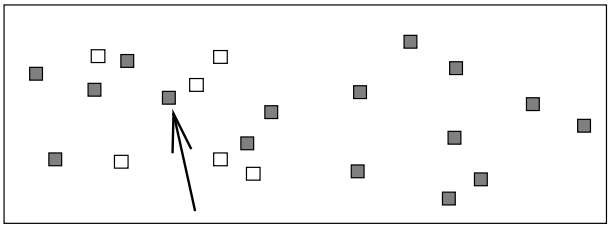

Figure 3. Selecting the closest sequence

Let us illustrate this by means of an example. Suppose the domain is composed of 82 sequences of lexemes corresponding each to an Aesop's fable. Then suppose that a beginner student was asked to provide an English text in order for the process to be initiated. The user model is composed of only this sequence of lexemes:

> My English is very basic. I know only a few verbs and a few nouns. I live in a small village in the mountains. I have a beautiful brown cat whose name is Felix. Last week, my cat caught a small bird and I was very sorry for the bird. He was injured. I tried to save it but I could not. The cat did not understand why I was unhappy. I like walking in the forest and in the mountains. I also like skiing in the winter. I would like to improve my English to be able to work abroad. I have a brother and a sister. My brother is young.

Running LSA, the closest domain sequence is the following:

> Long ago, the mice had a general council to consider what measures they could take to outwit their common enemy, the Cat. Some said this, and some said that; but at last a young mouse got up and said he had a proposal to make, which he thought would meet the case. "You will all agree," said he, "that our chief danger consists in the sly and treacherous manner in which the enemy approaches us. Now, if we could receive some signal of her approach, we could easily escape from her. I venture, therefore, to propose that a small bell be procured, and attached by a ribbon round the neck of the Cat. By this means we should always know when she was about, and could easily retire while she was in the neighborhood." This proposal met with general applause, until an old mouse got up and said: "That is all very well, but who is to bell the Cat?" The mice looked at one another and nobody spoke. Then the old mouse said: It is easy to propose impossible remedies.

It is hard to tell why this text ought to be the easiest for the student. A first answer would be to observe that several words of the fable occured already in the student's text (like cat, young, small, know, etc.). However, LSA is not limited to occurrence recognition: the mapping between domain and student's knowledge is more complex. A second answer is that the writer of the first text actually found that fable the easiest from a set of 10 randomly selected ones. The third answer is that LSA has been validated several times as a model of knowledge representation; however, experiments with many subjects need to be performed to validate that particular use of LSA.

Although the closest sequence could be considered the easiest by the student, it is probably not suited for learning because it is in fact too close to the student's knowledge.

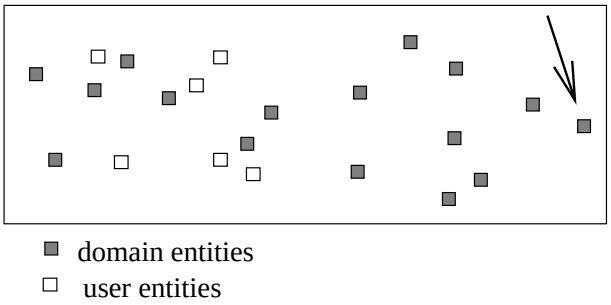

Figure 4. Selecting the farthest sequence

5.2. Selecting the farthest sequence

Another solution would be then to choose the farthest sequence (Figure 4). In our example, this would return::

> A Horse and an Ass were travelling together, the Horse prancing along in its fine trappings, the Ass carrying with difficulty the heavy weight in its panniers. "I wish I were you," sighed the Ass; "nothing to do and well fed, and all that fine harness upon you." Next day, however, there was a great battle, and the Horse was wounded to death in the final charge of the day. His friend, the Ass, happened to pass by shortly afterwards and found him on the point of death. "I was wrong," said the Ass: Better humble security than gilded danger.

That sequence was found quite hard to understand by our writer. Choosing the farthest sequence is therefore probably not appropriate for learning either, because it is too far from the student's knowledge.

5.3. Selecting the closest sequence among those that are far enough

None of the previous solutions being satisfactory, a solution would then be to ignore domain sequences that are too close to any of the student sequences. A zone is therefore defined around each student sequence and domain sequences inside these zones are not considered (we present a way of implementing that procedure in the next section). Then by using the same process described in the previous section, we select the closest sequence from the remaining ones. Figure 5 illustrates this selection.

The idea that learning is optimal when the stimuli is neither too close nor too far from the student's knowledge has been theorized by Vygotsky (1962) with the notion of *zone of proximal development*. He influenced Krashen (1988) who defined the *input hypothesis* as an explanation of how a second language is acquired: the learner improves

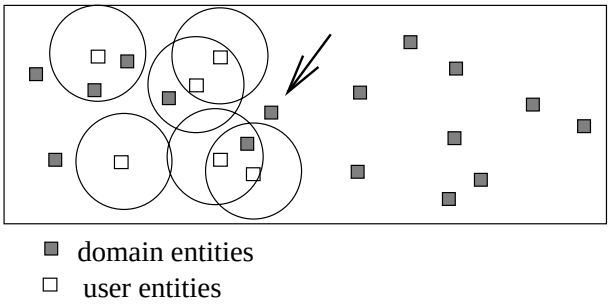

Figure 5. Selecting the next stimulus: the closest among those that are far enough

his/her linguistic competence when he receives second language 'input' which is one step beyond his/her current stage of linguistic competence.

6. Applications

We designed two systems along the previous theoretical ideas.

6.1. A system to help learning a language

We first designed a program in C, in the domain of language learning. It is based on the result previously mentioned that most of the words we know, we learned from reading. Therefore, the goal is, at each step, to find the most appropriate English text for French students to read in order to stretch the student subspace.

First, LSA is run to place all domain texts in a semantic space.We also added lots of English texts so that the accuracy of semantic proximities is better. However, these additionnal texts are not taken into account in the following procedure.

The systems works in the following way: it selects a text dynamically, presents it to the student, tests the comprehension, then selects another text, etc. The process is initialized with a text the student provides (it will also work if the student provides no text ; it will just take longer to reach a correct behavior of the system). At the beginning, the system does not know much about the student and therefore, the selection of the next text might not be optimal. However, after a time the user model is more and more precise and the choice of the system more accurate.

After each text is provided, the student is required to rate his/her comprehension on a 1 to 5 scale. The text is then added to the student model as well as its weight. This is used to compute the proximity between a domain text and a student text: the similarity provided

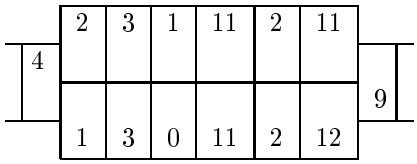

Figure 6. A state of the game of kalah

by LSA is multiplied by the weight. Therefore, texts that were well understood by the student play a more important role in the selection of the next text.

Improvements could be made by allowing the student to select words or parts of the texts that were not understood. This is going to be the object of our future work.

6.2. A system to help learning kalah

In the same way, we designed a program to help a student learn an African game called kalah. This program, written in C, can be viewed as a tutor: it plays in such a way that the student is driven towards a state which should be optimal for learning.

Kalah is played on a board composed of two rows of 6 pits. Each player owns a row of 6 pits as well as special pit called kalah. Pits initially contain 6 stones, and both kalahs are empty. Each player takes all the stones in any of his 6 pits, then spread them over the pits counter-clockwise, one stone per pit, including his kalah (but not the opponent's kalah). If the last stone lands in the kalah, the player has another turn. If the last stone lands in an empty pit, and the opponent has stones in the opposite pit, then all these stones go in the kalah. The goal is to get as many stones as possible in the kalah.

Lexemes are elements for describing a state. For instance, the lexeme `a3` indicates that there are 3 stones in the pit `a` (pits including kalahs are labelled from `a` to `n`).

A sequences of lexemes represents a state of the game. For instance the state shown in figure 6 is represented by the following sequence of lexemes: `a1 b3 c0 d11 e2 f12 g9 h11 i2 j11 k1 l3 m2 n4`.

A semantic space was built from 50,000 states automatically generated by two C programs playing together using a traditional minmax algorithm at a depth of 3. This semantic space therefore covers a large part of the kalah semantics.

The system plays against the student. Each time the student encounters a new configuration of the board, it is recorded into the space as a student entity. At each turn, the system looks for the different

possible moves. Each one results in a new state, that is a new sequence of lexemes. The system looks for the new sequence of lexemes which is close enough to the student model but not too close (we rely on the same procedure described earlier). Then it plays the corresponding move. For instance, in the previous figure, there are 6 possible moves, therefore 6 possible new states (the system plays the upper row):

1. `a2 b3 c0 d11 e2 f12 g9 h11 i2 j11 k1 l3 m0 n5`
2. `a2 b3 c0 d11 e2 f12 g9 h11 i2 j11 k1 l0 m3 n5`
3. `a1 b3 c0 d11 e2 f12 g9 h11 i2 j11 k0 l4 m2 n4`
4. `a2 b4 c1 d12 e3 f13 g9 h12 i2 j0 k2 l4 m3 n5`
5. `a1 b3 c0 d11 e2 f12 g9 h11 i0 j12 k2 l3 m2 n4`
6. `a2 b4 c1 d12 e3 f12 g9 h0 i3 j12 k2 l4 m3 n5`

States 1 and 2 give the student the opportunity to play one more move and to put the last stone in an empty hole (which would allow him to capture the opposite stone). State 4 shows to the student 13 stones in a hole, which is a special number because after going round the board the last stone falls necessarily in an empty hole. State 5 gives the student the possibility to play again. States 3 and 6 have nothing special.

According to the student model, state 5 is considered the most appropriate. Therefore move 5 will be played by the machine.

This system does not play optimally; indeed it sometimes plays badly since it is only concerned with driving the student towards an appropriate part of the semantics of the domain. As long as the student is aware that he is playing against a player with no winning behavior, we believe that this is no problem.

7. Conclusion

In this paper, we relied on a high-dimensional representation of the lexemes of a domain to build the framework of a tutoring system. The goal of this tutoring system is:

- to automatically detect lexeme misunderstandings ;
- to select the next stimulus to expose the student to in order for the learning to be optimal.

LSA has been used elsewhere for the design of a tutoring system (Wiemer-Hastings, 1999a; Wiemer-Hastings, 1999b). However, Autotutor is somehow different since LSA is not used to represent knowledge but rather to assess student productions in a tutoring dialogue. The strength of Autotutor is that the student can use natural language to interact with the system, without being limited to single words answers.

Our approach is based on a model that has been validated in the field of cognitive psychology. This point strikes us as being very important: a problem in the field of educational technology is that too many systems are built without strong theoretical cognitive fundations. The link between a psychological theory of learning and a technology was obvious in programmed teaching which was based on neobehaviourism. It was also the case with the theory of constructivism which founded the design of systems like microworlds. However, we all noticed that the arrival of multimedia capabilities sometimes lead to the design of systems that were not based on psychological theories of learning, but rather on the spectacular side of technology. The strength of LSA relies on a dual feature: it is both a tool for knowledge representation and a model of learning.

We believe that our approach could be useful in many areas as long as examples can be found and decomposed into elements that are the "words" of the domain. This decomposition is very important in order for the method to work. This can actually be a problem in some areas.

- Since LSA does not take into account the word order inside texts, decomposition should be such that the element order inside solutions should not matter. This is not necessarily a limitation with language: psychological experiments showed that the meaning of words can be derived independently from word order (Landauer et al., 1997)[5] However, this could be a problem in some domains, in particular if lexeme are actions, because the fact that `action1` occurs before `action2` might be important in the domain. One way to alleviate that problem could be that the solution consists of not only the various actions but also the order constraints between them. In that case, a solution would be:

  `action1 ... actionp 1precedes2 1precedes3....`

  One of our next tasks will be to study this problem.

- If the lexemes belong to a large lexis (for instance if the number of different lexemes is big) there will be too few co-occurrences and the method will not work. On the other hand, if the lexis is

[5] It does not mean that syntax plays no role. Its role could be to reduce the cognitive load in the semantic processing of text.

too small, there will be too few similarities and the method will not work either. In the domain of language, that size is several thousand. In the domain of tic-tac-toe, the number of different positions is 18 (9 for the `x`, 9 for the `o`). In the game of chess, it is several thousand. It is hard to evaluate an adequate size and more empirical work should be done.

Acknowledgements

We thank Susan Dumais and the Bellcore Labs for having allowed us to use and hack the code of the basic LSA programs. We are also grateful to Erica de Vries and Philippe Dessus for their comments on previous versions of this article.

References

Burgess, C. and K. Lund. Modelling Parsing Constraints with High-dimensional Context Space. *Language and Cognitive Processes*, 12(2/3):177–210, 1997.

Deerwester, C., S.T. Dumais, G.W. Furnas, T.K. Landauer and R. Harshmann. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.

Dumais, S.T., T.A. Letsche, M.L. Littman and T.K. Landauer. Automatic cross-language retrieval using Latent Semantic Indexing. In Hull D, Oard D, (Eds.), *1997 AAAI Symposium on Cross–Language Text and Speech Retrieval*. American Association for Artificial Intelligence, 1997.

Foltz, P.W. and S.T. Dumais. Personalized Information Delivery: An Analysis of Information Filtering Methods. *Communications of the ACM* 35(12): 51–60, 1992.

Foltz, P.W. Latent Semantic Analysis for text-based research. *Behavior Research Methods, Instruments, & Computers* 28(2):197–202, 1996.

Foltz, P.W., W. Kintsch and T.K. Landauer. The Measurement of Textual Coherence with Latent Semantic Analysis. *Discourse Processes*, 25:285–307, 1998.

Kintsch, E., Steinhart D., Stahl G. and the LSA Research Group. Developing Summarization Skills through the Use of LSA-based Feedback. *Interactive Learning Environments*, to appear.

Krashen, S.D. *Second Language Acquisition and Second Language Learning*, Prentice-Hall International, 1988.

Landauer, T.K. and S.T. Dumais. A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction and Representation of Knowledge. *Psychological Review* 104(2):211–240, 1997.

Landauer, T.K., D. Laham, B. Rehder and M.E. Schreiner. How Well Can Passage Meaning be Derived without Using Word Order? A Comparison of Latent Semantic Analysis and Humans. In Shafto, M.G., Langley, P. (Eds), *Proceedings of the 19th annual meeting of the Cognitive Science Society* 412–417, Mawhwah, NJ:Erlbaum, 1997.

Lemaire, B. Models of High-dimensional Semantic Spaces. *Proceedings of the 4th International Workshop on MultiStrategy Learning (MSL'98)*, 1998.

Lemaire, B. Tutoring Systems based on Latent Semantic Analysis In S.P. Lajoie and M. Vivet (Eds), *Artificial Intelligence in Education* (proceedings of the AIED'99 Conference), IOS Press, 527-534, 1999.

Lemaire, B. and P. Dessus. A System to Assess the Semantic Content of Student Essays. *Journal of Educational Computing Research*, in press.

Lund K. and C. Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Method, Instruments, & Computers* 28(2), 203-208, 1996.

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

Saussure, F. *Saussure's Third Course of Lectures in General Linguistics*. Pergamon Press, 1993.

Vygotsky, L.S., *Thought and Language*, Cambridge, M.I.T. Press, 1962.

Wenger, E. *Artificial Intelligence and Tutoring Systems*. Morgan Kaufman, 1987

Wiemer-Hastings, P., K. Wiemer-Hastings and A.C. Graesser. Improving an intelligent tutor's comprehension of students with Latent Semantic Analysis. In S.P. Lajoie and M. Vivet (Eds), *Artificial Intelligence in Education* (proceedings of the AIED'99 Conference), IOS Press, 535-542, 1999.

Wiemer-Hastings, P., K. Wiemer-Hastings and A. Graesser. Approximate Natural Language Understanding for an Intelligent Tutor In *Proceedings of the 12th Florida Artificial Intelligence Research Symposium (FLAIRS'99)*, 1999.

Wolfe, M.B., Schreiner M.E., Rehder B. and Laham D. Learning from text: Matching readers and texts by Latent Semantic Analysis. *Discourse Processes*, 25:2-3, 309-336, 1998.