Behavior Research Methods
2006, 38 (4), 628-637

A computational model for simulating text comprehension

BENOÎT LEMAIRE
University of Grenoble, Grenoble, France

GUY DENHIÈRE
CNRS and University of Provence, Marseille, France

CÉDRICK BELLISSENS
University of Memphis, Memphis, Tennessee

and

SANDRA JHEAN-LAROSE
University of Paris VIII and IUFM, Paris, France

In the present article, we outline the architecture of a computer program for simulating the process by which humans comprehend texts. The program is based on psycholinguistic theories about human memory and text comprehension processes, such as the construction–integration model (Kintsch, 1998), the latent semantic analysis theory of knowledge representation (Landauer & Dumais, 1997), and the predication algorithms (Kintsch, 2001; Lemaire & Bianco, 2003), and it is intended to help psycholinguists investigate the way humans comprehend texts.

This article describes the architecture of a computer program that aims to simulate the process by which humans comprehend texts—that is, construct a coherent representation of the meaning of the text by processing all sentences in turn. This program is based on psycholinguistic theories about human memory and text comprehension processes—namely, the construction–integration model (Kintsch, 1998), the latent semantic analysis theory of knowledge representation (Landauer & Dumais, 1997), and the predication algorithms (Kintsch, 2001; Lemaire & Bianco, 2003). It is not a natural language processing tool, although this community may benefit from its ideas. Neither is it the best program for automatically analyzing texts. Rather, it is designed to mimic—as closely as possible—human beings (especially children) reading texts. It was intended to help psycholinguists implement theories, test ideas, and identify relevant cognitive variables. For these reasons, this program is largely modular and parameterizable so that researchers can use it as a tool for exploring the cognitive processes underlying human text comprehension.

It is worth noting that, for the sake of comprehension, we will not present the full architecture at one go; rather, we will first describe the core of the architecture, then different modules that aim to improve the initial system. The first module is a model of semantic memory.

We thank Philippe Dessus for his comments on a previous version of this article. Address correspondence to B. Lemaire, Laboratoire TIMC-IMAG (CNRS UMR 5525), Faculté de Médecine, 38706 La Tronche Cedex, France (e-mail: benoit.lemaire@imag.fr).

LSA: A MODEL OF SEMANTIC MEMORY

Principle

As major models of text comprehension (e.g., construction–integration [Kintsch, 1988], landscape model [van den Broek, Risden, Fletcher, & Thurlow, 1996], and resonance model [Gerrig & McKoon, 1998; Myers & O'Brien, 1998]) have shown, comprehending a text cannot be done with only the information present in the text (Caillies, Denhière, & Jhean-Larose, 1999; McNamara & Kintsch, 1996; Rizzella & O'Brien, 2002). Readers need to rely on their knowledge of the world. Actually, cognitive theories of text comprehension assert that readers would automatically activate concepts while reading (Kintsch, 1998; van den Broek, Young, Tzeng, & Linderholm, 1999). Therefore, a simulation has to be based on a computational model of semantic memory that would be able to provide semantic associates for any word, thus simulating the automatic activation of concepts in memory (Caillies & Denhière, 2001; Tapiero & Denhière, 1995). Associates are obviously not predefined; rather, they depend on the reader's knowledge. In order to simulate text comprehension for different kinds of readers—expert or novice in a given domain, adults or children of various ages—we could not rely on a predefined set of associates for every word (not to mention the fact that such association norms do not exist for all words) (Caillies, Denhière, & Kintsch, 2002). Ideally, we would need to construct word similarities from the same kind of stimuli humans experience. That way, we would get word similarities for medical experts, an average teenager, a 7-year-old child, and so on. Since the perceptual experience on which humans rely cannot yet be captured by a computational model, we restricted our input to the linguistic experience, which, albeit not perfect, appears to play an important role in the construction of word meaning (Landauer, 2002).

We used latent semantic analysis (LSA; Deerwester, Dumais, Furnas, Landauer, & Harshman, 1990; Landauer, 1998; Landauer & Dumais, 1997), a computational model of word similarities that is based on the automatic analysis of huge corpora and roughly reproduces the kind of text people have been exposed to. The underlying idea is that the meaning of words can be inferred from the contexts in which these words occur in raw texts, provided that enough data are available (Landauer, 2002). This is similar to what humans do: It seems that we learn most of the words we know by reading (Glenberg & Robertson, 2000; Landauer & Dumais, 1997). This is because most words appear almost only in written form and because direct instruction seems to play a limited role. Therefore, we learn the meaning of words mainly from raw texts, by mentally constructing their meaning through repeated exposure to appropriate contexts (Denhière, Lemaire, Bellissens, & Jhean-Larose, 2007; Kintsch, 2007).

LSA analyzes the co-occurrence of words in large corpora to draw semantic similarities. In order to facilitate the measurement of similarities between words, LSA relies on very simple structures to represent word meanings: All words are represented as high-dimensional vectors. The meaning of a word is not defined, per se; rather, it is determined by its relationships with all others. For instance, instead of defining the meaning of *bicycle* in an absolute manner (e.g., by its properties, function, or role, as in semantic networks), it is defined by its degree of association to other words (e.g., very close to *bike*, close to *pedals*, *ride*, *wheel*, but far from *duck*, *eat*). Once again, this semantic information can be drawn from raw texts.

The problem is how to go from these raw texts to a formal representation of word meanings. One way to tackle this would be to rely on direct co-occurrences within a given unit of context. A usual unit is the paragraph, which is both computationally easy to identify and of reasonable size. We would say that

R1: *Words are similar if they occur in the same paragraphs*.

Therefore, we would count the number of occurrences of each word in each paragraph. Suppose we rely on a 5,000 paragraph corpus. Each word would be represented by 5,000 values—that is, by a 5,000 dimension vector. For instance,

avalanche: (0,1,0,0,0,0,1,0,2,0,0,0,0,0,0,1,1,0,1,0,1, 0,0,0,0,0,0 . . .)

snow: (0,2,0,0,0,0,0,0,1,1,0,0,0,0,0,0,2,1,1,0,1,0,0, 0,0,0,0 . . .)

This means that the word *avalanche* appears once in the 2nd paragraph, once in the 7th, twice in the 9th, and so on. One can see that, given the previous rule, both words are quite similar: They co-occur quite often. A simple cosine between the two vectors can measure the degree of similarity. However, this rule does not work well (Landauer, 2002; Perfetti, 1998): Two words could be considered similar even though they do not co-occur. For instance, Burgess and Lund (1997) mentioned two words—*road* and *street*—that almost never co-occur in their huge corpus, even though they are almost synonyms. In a 24 million word French corpus from the daily newspaper *Le Monde* in 1999, we found 131 occurrences of *Internet*, 94 occurrences of *Web*, but no co-occurrences of these two words at all. However, both words are strongly associated. The reason why two words are associated in spite of no co-occurrences between them could be that both co-occur with a third word. For instance, if you mentally construct a new association between *computer* and *quantum* from a set of texts you have read, you will probably also construct an association between *microprocessor* or *quantum*—even though they might not co-occur—because of the existing strong association between *computer* and *microprocessor*. The relationship between *computer* and *quantum* is called a *second-order co-occurrence*. Psycholinguistic research on mediated priming has shown that the association between two words can be made through a third one (Livesay & Burgess, 1997; Lowe & McDonald, 2000), even if the explanation for this phenomenon is debatable (Chwilla & Kolk, 2002). Let us go a little farther. Suppose that the association between *computer* and *quantum* is also a second-order association, because of another word that co-occurs with both words—for example, *science*. In that case, *microprocessor* and *quantum* are said to be *third-order* co-occurring elements. In the same way, we can define fourth-order co-occurrences, fifth-order co-occurrences, and so on. Kontostathis and Pottenger (2002) analyzed such connectivity paths in several corpora and found the existence of these high-order co-occurrences.

French and Labiouse (2002) thought that the previous rule might still have worked for synonyms. This is because writers tend to use synonyms rather than repeat words. However, defining semantic similarity only from direct co-occurrence is probably a serious restriction. Therefore, another rule would be

R1*: *Words are similar if they occur in similar paragraphs*.

This is a much better rule. Consider the following two paragraphs:

Cycling is a very pleasant sport. It helps maintain good health.

To keep fit, you could practice biking. It is very pleasant and good for your body.

Here *cycling* and *biking* appear in similar paragraphs. If these words are then repeated over a large corpus, it would be reasonable to consider them similar, even if they never co-occurred within a paragraph. Now we need to define paragraph similarity. We could say that two paragraphs would be similar if they shared words, but that would be restrictive. As illustrated in the previous example, two paragraphs should be considered similar even though they do not have words in common (functional words are usually not taken into account). Therefore, the rule is

R2: *Paragraphs are similar if they contain similar words.*

Rules 1* and 2 constitute a circularity, but this can be solved by a specific mathematical procedure called *singular value decomposition*, which is applied to the occurrence matrix. This is exactly what LSA does. LSA consists in reducing the huge dimensionality of direct word co-occurrences to its best *N* dimensions. All words are then represented as *N*-dimensional vectors. Empirical tests have shown that performances are maximal for *N* around 300 for the whole general English language (Bellegarda, 2000; Landauer, Foltz, & Laham, 1998), but this value can be smaller for specific domains (Dumais, 2003). We will not describe the mathematical procedure, which is presented in detail elsewhere (Deerwester et al., 1990; Landauer, 1998). The fact that word meanings are represented as vectors has two consequences. First, it is straightforward to compute the semantic similarity between words—usually the cosine between the corresponding vectors, although other similarity measures can be used. Examples of semantic similarities between words from a 12.6 million word corpus are (Landauer, 2002):

cosine (*doctor*, *physician*) = .61
cosine (*red*, *orange*) = .64

As has been done in many other studies in the literature, we checked whether LSA can be considered as a good model of semantic memory. We wanted LSA to provide good associates for any given word, in order to simulate the mental activation of concepts that occurs in humans when they process a word. Because of its vector representation, LSA can easily return the closest neighbors of a given word.

Corpus

Actually, LSA by itself is useless. It must be applied to a corpus. We have several corpora, but our most elaborate one is a child corpus that we carefully designed in order to reproduce—as closely as possible—the kinds of texts children are exposed to (Denhière & Lemaire, 2004). We controlled the amount and nature of texts, leading to a 3.2 million word corpus that was composed of stories and tales for children (~1.6 million words), child productions (~800,000 words), reading textbooks (~400,000 words), and a child encyclopedia (~400,000 words).

We tested whether the closest neighbors of a given word would correspond to the words that were activated in memory by children. We relied on verbal association norms (de la Haye, 2003) that were defined in the following way: Two hundred inducing words (144 nouns, 28 verbs, and 28 adjectives) were proposed to children from the ages of 9 to 11. For each word, participants had to provide the first corresponding word that came to their minds. This resulted in a list of words ranked by frequency. For instance, given the word *cartable* (*satchel*), 9-year-old children had the following results:

Three best-ranked words:

école (*school*): 51%
sac (*bag*): 12%
affaires (*stuff*): 6%

Three worst-ranked words:

classe (*class*): 1%
sacoche (*satchel*): 1%
vieux (*old*): 1%

This means that 51% of the children responded with the word *école* (*school*) when given the word *cartable* (*satchel*). The two words are therefore strongly associated for 9-year-old children. These association values were compared with the LSA cosine between word vectors. We selected the three best-ranked words, as well as the three worst-ranked words (as we did in the previous example). We then measured the cosines between the inducing word and the first-ranked association, the second-ranked association, and the third-ranked association, as well as the mean cosine between the inducing word and the 3 worst-ranked associations. Results are presented in Table 1.

Student tests showed that all differences were significant ($p < .03$). This means that our semantic space was not only able to distinguish between the strong and weak associations, but could also discriminate the first ranked from the second ranked and the latter from the third ranked.

The measure of correlation with human data was also significant [$r(1184) = .39, p < .001$]. Actually, two factors may have reduced this correlation. First, although we tried to mimic what a child had been exposed to, we could not control all word frequencies within the corpus. Therefore, some words might have occurred with a low frequency in the corpus, leading to an inaccurate semantic representation. When the previous comparison was performed on the 20% most frequent words, the correlation was much higher [$r(234) = .57, p < .001$].

The second notable factor is the participant agreement. When most children provided the same answer to an inducing word, there was a high agreement, which means that both words were strongly associated. However, there were cases when there was almost no agreement. For instance, the three first responses to the word *bruit* (*noise*) were *crier* (*to shout*) (9%), *entendre* (*to hear*) (7%), and *silence* (*silence*) (6%). It is not surprising that the model corresponded better to the children's data in the case of a high agreement, since this denotes a strong association that should be reflected in the corpus. In order to select answers with stronger agreement, we measured their entropy using the following formula:

$$\text{entropy(item)} = \sum \text{freq(answer)} \cdot \log[1/\text{freq(answer)}].$$

A low entropy corresponds to a high agreement, and vice versa. When we selected the 20% of items with the low-

Table 1
Mean Cosine Between Inducing Word and Various Associated Words for 9-Year-Old Children

	Word Ranking	Mean Cosine With Inducing Word
Best	first	.26
	second	.23
	third	.19
Worst	third	.11

est entropy, the correlation also increased [$r(234) = .48$, $p < .001$].

All these results show that the degree of association between words that were defined by the cosine measure within the semantic space seemed to correspond quite well with the children's judgments of association. LSA—applied to our child corpus—is an acceptable model of semantic memory.

In order to simulate adult comprehension, we built another semantic space based on the previous child corpus, as well as a newspaper corpus and a literature corpus. This adult corpus, therefore, is composed of about 13 million words: A 3 million word children's corpus, a 5 million word corpus from the French daily newspaper *Le Monde*, plus a 5 million word corpus composed of French novels. This corpus was processed using LSA, and a 300-dimension semantic space was built. This semantic space was used to analyze the example test that will be discussed in a later section.

A MODEL OF TEXT COMPREHENSION

Now that we have a good model of semantic memory, we need a model of text comprehension as well. That model should describe the process by which a set of sentences is transformed into a coherent representation of the overall meaning of the text. The theoretical model we are using is the construction–integration model (Kintsch, 1998). Discourse comprehension is viewed as an iterative two-step process. First, the current proposition (or set of propositions) leads to the construction of a network of concepts that either belong to the proposition or are activated from semantic memory. This network is added to another network called the *macrostructure*, which results from the analysis of the prior part of the text and represents the main information so far. Second, the integration step selects the relevant concepts from this network by means of a spreading activation mechanism, leading to the new macrostructure. The process is repeated until the whole text is processed.

We will now present our operationalization of that model in a computer program. Consider a text composed of these two sentences:

The bee is sucking nectar from a flower. Then it brings the nectar back to the hive to be turned into honey.

The main process of text comprehension occurs within the specific component called *working memory*.[1] This component contains key elements of the sentences that have previously been processed, as well as the elements of the current sentence. As we mentioned previously, the reader would also activate concepts from semantic memory. For instance, the word *bee* would activate words such as *honey*, *hive*, or *sting*. Three kinds of elements, therefore, are gathered in working memory: the previous ones, the current ones, and a set of associates. Since not all of these are coherent with the context, the integration step selects the most relevant ones—that is, those that are loosely connected to the others. For instance, *sting* is not strongly associated to most of the other words and must be dismissed. This integration step is performed by means of a spreading activation mechanism, which is run until the system becomes stable.

Working memory is thus continuously updated as the text is processed, while containing the main information from what has already been processed. It is worth noting that some of these words are not part of the text; like *honey*, they are inferences of a sort that readers make by means of their semantic memory.

What is true for words is also true for propositions, or subsets of sentences. For instance, the previous text contains the following propositions:

P1: *sucking* (*bee*, *nectar*, *flower*)
P2: *bring* (*bee*, *nectar*, *hive*)
P3: *turn* (*nectar*, *honey*)
P4: *for* (*P2*, *P3*)

A proposition may also activate associates, can be propagated as a key feature of the overall meaning, and can occasionally be ruled out if it becomes secondary.

To summarize, each proposition is processed in turn. Inferences are gathered from semantic memory. An integration of this new information and previous information is realized in order to reach a new state of working memory. Figure 1 displays the flow of information for each proposition (episodic memory will be presented later).

A French translation of the previous example was simulated by our program (without taking into account the predication algorithm and the episodic memory, which will be presented in the next sections), using the previous French model of semantic memory. We now present the English translation. The first proposition was *sucking* (*bee*, *nectar*, *flower*). It activated the following elements:

insect, *larva*, *fly*, *hive*, *honey*, *wasp*, *buzz*, *bouquet*, *violet*, *petal*, *gather*, *blossom*

Semantic similarities between all pairs of words were then computed, leading to a large semantic network. The most relevant elements (those that were the most coherent with all others) were selected by the integration step. The working memory then contained the following elements (as well as their activation values):

sucking (*bee*, *nectar*, *flower*)	1.000
bee	.903
flower	.852
hive	.778
bouquet	.677
buzz	.634
honey	.615
petal	.607
wasp	.606
violet	.605

The second group of propositions was then added to working memory. It was *bring* (*bee*, *nectar*, *hive*) and *turn* (*nectar*, *honey*). Semantic similarities between all of these words and propositions were computed. Since both propositions occurred in the same input stream, a 1.0 link was created between the last two propositions to represent

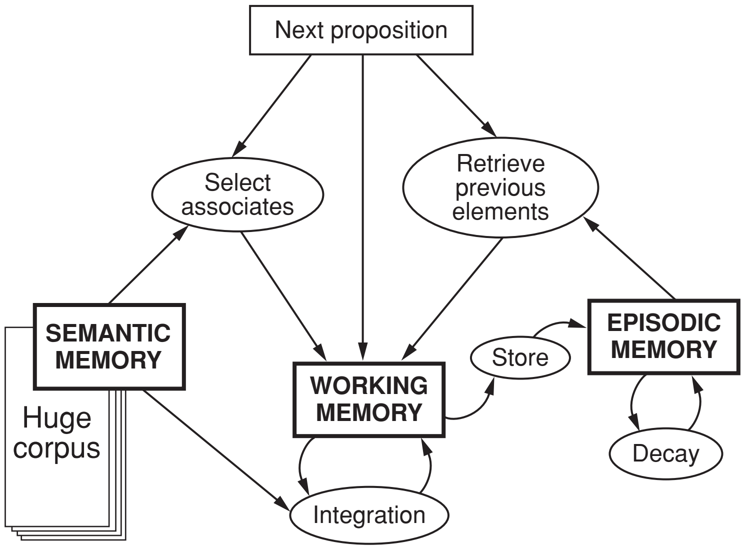

Figure 1: Information flow of the comprehension model.

their strong connection in the text. The first one activated the following elements:

worker, *hive*, *honey*, *wasp*, *buzz*, *fly*

The second one activated

take, *mineral*, *meaning*, *sugar*, *living*, *hive*, *bee*, *bear*, *bear cub*, *pheasant*

Together with the previously activated elements, this led to the following set of elements:

sucking (*bee*, *nectar*, *flower*), *bee*, *flower*, *hive*, *bouquet*, *buzz*, *honey*, *petal*, *wasp*, *violet*, *bring* (*bee*, *nectar*, *hive*), *worker*, *fly*, *turn* (*nectar*, *honey*), *take*, *mineral*, *meaning*, *sugar*, *living*, *bear*, *bear cub*, *pheasant*

The most activated elements from this set were selected. The working memory was then as follows:

sucking (*bee*, *nectar*, *flower*)	1.000
bring (*bee*, *nectar*, *hive*)	.997
bee	.949
hive	.868
turn (*nectar*, *honey*)	.813
honey	.805
buzz	.612

The next set of propositions was considered, and its elements and associates were added to working memory. After each new set of propositions was analyzed, working memory represented a sort of synthesis of the information processed so far.

EPISODIC MEMORY

We now present a new structure, episodic memory. Prior elements removed from working memory are meant to be no longer necessary; however, they are still kept in a specific memory that keeps track of all the elements that have appeared in working memory. These can even be retrieved from working memory in case they become relevant with respect to the text content. The elements are stored with an activation value, which may vary over time, depending on whether or not they appear again in working memory. A decay function tends to lower these values over time, thus simulating a sort of forgetting mechanism.

From Working Memory to Episodic Memory

Episodic memory is defined by means of three functions that have the goal of determining the activation values (from 0 to 1). These functions are applied every time an element of working memory is stored in episodic memory, and are as follows.

The first function indicates the new value of a concept that did not previously exist in episodic memory. By default, the new value is the activation value of the concept in the working memory.

The second function defines the new value of a concept that was already in episodic memory. In that case, the new value should be higher than both existing values, because of the conjunction of the two memory traces. By default, the new value is valueWM + valueEM · (1 − valueWM).

The third one is a decay function that indicates how to lower all activation values over time. By default, all values are changed to 90% of their original values after each construction–integration cycle.

From Episodic Memory to Working Memory

During the construction phase, episodic memory can also provide elements that are added to working memory if they are close to the text elements being processed. This is similar to the inference mechanism that gathered elements from semantic memory.

The idea is that all episodic memory elements that are similar enough to a concept of the current proposition—and that have a high enough activation value—are copied back into working memory. The two thresholds that govern this collection of elements are obviously parameterizable. This would be the case of a text that would present a topic *X*, then shift to topic *Y*—leading to the removal of concepts related to *X* in working memory—then go back to topic *X*. The current mechanism would then retrieve *X*-related concepts from episodic memory in order to simulate the fact that concepts can be linked in a text even though they do not necessarily follow each other.

At the end of the text processing, episodic memory contains all the propositions from the text and has an indication of their importance. This model of the way in which the main information has been cognitively selected can be tested and compared with human data. In addition, since every state of episodic memory is memorized by the program, the evolution of activation values can be traced. The decay function tends to decrease activation values of unused elements over time; however, when an element appears once again in working memory—whether because it occurs in the text or because it has been called back by a similar element—its activation value rises. Evolution of activation values in episodic memory is not linear and depends on the propositions being processed. Once again, the fact that this structure is automatically produced for any kind of text is valuable for researchers willing to test and refine the model. Episodic memory is presented in Figure 1.

A MODEL OF PREDICATION

We now present an improvement on the previous models. When a word is processed, its neighbors are activated from semantic memory, as we mentioned earlier. The same occurs for propositions: Neighbors of all words of the proposition should be activated. For instance, when you read the sentence *the plane flies to Paris*, you mentally gather associates for *plane*, *flies*, and *Paris*. However, only the neighbors of the predicate that are associated to the context need to be considered: You select associates such as *airport* or *sky*, but not *escape* or *fear*, because—although they are close neighbors of *fly*—they are not related to the arguments. Kintsch (2001) has shown that the LSA model can be used to provide a good semantic representation of a predicate–argument expression, if the specific role of the predicate is taken into account.

The basic LSA representation does not make any distinction between A(B) and B(A), because the compositionality consists only of adding vectors: The vector representing a set of words is just the sum of the vectors of all words. *This child is a sportsman* has the exact same representation as does *This sportsman is a child*, which is particularly a problem for dealing with metaphors (Kintsch, 2000). To solve that problem, Kintsch suggested constructing a network composed of the predicate, the argument, and a fixed number of neighbors of the predicate, and applying the integration method to select only the neighbors that are associated with the predicate. Kintsch (2007) provided a little illustrative example with only three neighbors. Suppose there are three neighbors of *run*: *come*, *hopped*, and *down.* The sentence *the horse runs* will lead to a network composed of *horse*, *run*, *come*, *hopped*, and *down*. *Come* will be the only neighbor activated, because it is similar to both *horse* and *run*. On the contrary, in the sentence *the color runs*, only the neighbor *down* will be selected.

The representation of the predicate–argument expression, therefore, is not just predicate + argument but is predicate + argument + neighbor^1 + · · · + neighbor^n. We are not interested in the vector representation, but in the neighbors. Kintsch's (2007) algorithm can be a good starting point for our purpose. The problem is that this algorithm requires a number of neighbors to be set beforehand: 20 for usual predicate–argument relations, but up to 500 for some metaphors, according to Kintsch's experiments. Since the nature of the predicate–argument relation cannot be stated automatically, we had to modify this predication algorithm to make it incremental (Lemaire & Bianco, 2003). This modified version is included in the present comprehension program. Basically, if the input indicates which word is the predicate and which words are the arguments, the predication algorithm is used. It scans all neighbors of the predicate (using the model of semantic memory described earlier) until it finds three (or any other value of that parameter) of them that are similar enough (above a parameterizable threshold) to any of the arguments.

For instance, in our favorite French semantic space, the closest neighbors of the predicate *voler* (*to fly*) are the following:

ailes (*wings*)
oiseau (*bird*)
vole (*flies*)
plumes (*feather*)
oiseaux (*birds*)
aigle (*eagle*)
vol (*flight*)
. . .

When the input is *voler* (*avion*) [*fly* (*plane*)], the following words are selected because they are also similar to *plane*: *ailes* (*wings*), *vole* (*flies*), and *vol* (*flight*). However, when the input is *voler* (*oiseau*) [*fly* (*bird*)], the selected words are *ailes* (*wings*), *vole* (*flies*), and *plumes* (*feathers*).[2] The last version of the system includes this algorithm.

The associates of a proposition, therefore, are the associates of the predicate according to this algorithm, as well as the associates for all arguments. For instance, the proposition *fly* (*plane*) would activate not only *wings*, *flies*, and *flight*, but also *pilot*, *take off*, and *passengers*. The proposition *fly* (*bird*), rather, would activate *wings*, *flies*, and *feather* as well as *wings*, *bill*, and *plumage*.

A FULL EXAMPLE

We now present a full example. Suppose we want to simulate the comprehension of the following text:

Un bûcheron se promenait dans la forêt lorsqu'il vit une lumière. Des arbres brûlaient. Le bûcheron but l'eau de sa gourde et la cracha sur le feu. Le feu s'éteignit.

—which translates to

A woodcutter was walking in the forest when he noticed a light. Trees were burning. The woodcutter drank water from his flask and spit on the fire. The fire went out.

Since we intend to illustrate the predication algorithm in this example, we need to split the sentences into propositions and indicate which word in each proposition is the predicate. This cannot be done automatically for the moment (however, the model can be run automatically if the predication algorithm is not used). Propositions are represented as sequences of words whose predicate is in the first position. Inputs are therefore:

1. *walk/woodcutter/forest*; *notice/woodcutter/light*
2. *burn/trees*
3. *drink/woodcutter/water/flask*; *spit/woodcutter/fire*
4. *go out/fire* (*go out* is only one word in French)

A translation of the output of the program is shown in the shaded box at the top right of this page.

Words such as *stroll*, *oak*, or *glade* are now part of working memory, although they have not explicitly been mentioned in the sentence. Unrelated words such as *objects* or *shine* are ruled out from working memory, since their activation values are below the threshold.

The second sentence is analyzed next. As the reader will notice in the shaded box at the bottom right of this page, episodic memory elements close to the current input can be retrieved.

Only two propositions are kept in working memory; the second one (*notice/woodcutter/light*) disappears. The third sentence is analyzed next in the shaded box at the top left of the next page.

Four propositions and several related words (that are either part of the text, such as *forest*, or not part of the text, such as *trees* or *flames*) are in working memory. The last sentence is analyzed next in the shaded box at the bottom right of the next page.

At the end of the text, working memory contains the five main propositions and several related words. The model works fairly well, since all related words are really coherent with the context. This is due to two factors: the semantic memory model (LSA), which mostly retrieves relevant words, and the integration module, which rules out the possible remaining irrelevant words.

In addition to the last state of the working memory, our program provides the activation values of all words and propositions for each cycle (Table 2). For instance, the activation value of the proposition *notice*/*woodcutter*/*light* is .693 at the end of Cycle 1. It increases to .750 at the end of Cycle 2 and then decreases afterward. This data compares to the output of the landscape model (Linderholm, Virtue, Tzeng, & van den Broek, 2004), in which the activation value of concepts can be traced from proposition to proposition. The main difference, however, is that our system is based on a knowledge model (semantic memory): It can retrieve concepts that are not in the text and can automati-

```
*** SIMULATION OF TEXT COMPREHENSION (version 1.6.2) ***
Input? walk/woodcutter/forest notice/woodcutter/light
--------------------------
"walk/woodcutter/forest" added to working memory.
Looking for neighbors of walk:
   1. stroll (0.68) close to woodcutter and forest
   2. meet (0.60) close to woodcutter and forest
   3. pick (0.60) close to woodcutter and forest
woodcutter added to working memory. Looking for neighbors:
   1. ax (0.57)
   2. forest (0.53)
      firewood: too rare (.77 > .72)
   3. cottage (0.51)
forest added to working memory. Looking for neighbors:
   1. glade (0.77)
   2. oak (0.75)
   3. wood (0.74)
--------------------------
"notice/woodcutter/light" added to working memory.
Looking for neighbors of notice:
   1. objects (0.61) close to light
   2. watch (0.57) close to light
      commonly: too far from woodcutter and light
   3. area (0.56) close to light
woodcutter added to working memory. Looking for neighbors:
   previously done
light added to working memory. Looking for neighbors:
   1. luminous (0.80)
   2. rays (0.78)
   3. shine (0.69)
--------------------------
Constructing the 21x21 matrix...
Integrating... (9 cycles)
Activated nodes: walk/woodcutter/forest(1.00) forest(.891)
stroll(.887) oak(.868) glade(.837) woodcutter(.816) wood(.772)
notice/woodcutter/light (.770) pick(.748)
```

```
Input? burn/trees
"burn/trees" added to working memory.
   "rays" recoverable from episodic memory but too low (.450 < .75)
   "light" recoverable from episodic memory but too low (.500 < .75)
   "ax" recoverable from episodic memory but too low (.533 < .75)
   "walk" recovered from episodic memory by "trees". Added to WM.
   "oak" recovered from episodic memory by "trees". Added to WM.
Looking for neighbors of burn:
   1. burns: too far from trees
      fire: too far from trees
      burning: too far from trees
   1. heat (0.56) close to trees
      extinguish: too far from trees
      steam: too far from trees
   2. flames (0.54) close to trees
   3. burned (0.53) close to trees
trees added to working memory. Looking for neighbors:
   1. branches (0.90)
   2. trunks (0.87)
   3. leaves (0.82)
--------------------------
Constructing the 22x22 matrix...
Integrating... (6 cycles)
Activated nodes: burn/trees(1.00) walk/woodcutter/forest(.981)
trees(.920) forest(.900) trunks(.898) branches(.893) oak(.868)
wood(.806) glade(.791) pick(.766) leaves(.752)
```

```
Input? drink/woodcutter/water/flask spit/woodcutter/fire
"drink/woodcutter/water/flask" added to working memory.
  "walk" recovered from episodic memory by "drink". Added to WM.
  "stroll" recoverable from episodic memory but too low (.539 < .75)
  "ax" recoverable from episodic memory but too low (.480 < .75)
  "cottage" recoverable from episodic memory but too low (.474 < .75)
  "woodcutter" recovered from episodic memory by "flask". Added
  to WM.
  "flames" recoverable from episodic memory but too low (.374 < .75)
Looking for neighbors of drink:
  1. drinks (0.74) close to water and flask
  2. hot (0.74) close to water and flask
  3. drank (0.68) close to water and flask
woodcutter added to working memory. Looking for neighbors:
  1. ax (0.57)
  2. forest (0.53)
     firewood: too rare (.77 > .72)
  3. cottage (0.51)
water added to working memory. Looking for neighbors:
  shore: too rare (.94 > .72)
  1. drinkable (0.88)
     rat: too rare (.72 > .72)
  2. faucet (0.84)
  3. bucket (0.79)
flask added to working memory. Looking for neighbors:
     nibbling: too rare (.90 > .72)
  1. left (0.49)
  2. witch (0.49)
  3. potion (0.49)
-------------------------
"spit/woodcutter/fire" added to working memory.
  "burn" recoverable from episodic memory but too low (.424 < .75)
  "heat" recoverable from episodic memory but too low (.291 < .75)
  "flames" recoverable from episodic memory but too low (.374 < .75)
  "walk" recovered from episodic memory by "woodcutter". Added
  to WM.
  "ax" recoverable from episodic memory but too low (.480 < .75)
  "burned" recoverable from episodic memory but too low (.411 < .75)
  "burn" recoverable from episodic memory but too low (.424 < .75)
Looking for neighbors of spit:
  1. inhale (0.65) close to fire
  2. lukewarm (0.63) close to fire
     bleed: too far from woodcutter and fire
  3. spits (0.59) close to fire
woodcutter added to working memory. Looking for neighbors:
     previously done
fire added to working memory. Looking for neighbors:
  1. flames (0.71)
  2. burn (0.69)
  3. warm (0.65)
-------------------------
Constructing the 37x37 matrix...
Integrating... (8 cycles)
Activated nodes: walk/woodcutter/forest(1.00)
drink/woodcutter/water/flask(.956) forest(.908) wood(.903) oak(.887)
drink(.876) woodcutter(.855) trunks(.855) pick(.853) flask(.847)
burn/trees(.840) spit/woodcutter/fire(.834) glade(.834) branches(.796)
trees(.789) walk(.789) bucket(.770) hot(.754) drinks(.711) potion(.710)
flames(.704)
```

cally draw connections between concepts on the basis of their semantic similarities.

PARAMETERS

The program relies on 19 parameters, but many simulations have allowed us to identify good default values for most of them. This section describes the most important parameters.

Word Relevance

In LSA, weights are attached to words in order to indicate the knowledge that LSA has about words. This knowledge is dependent on the word frequency (LSA has better knowledge of words that occurred frequently in the corpus) and the context variability (LSA has better knowledge of words that occur in limited contexts than of words that appear in a large variety of contexts). Two parameters are used to rule out words that have high frequency but occur in a large number of contexts (like *the* or *and*), as well as words that are too rare.

Construction Phase

The number of neighbors is a parameter. The semantic memory model can also be modified. LSA is the default model, but others—like ICAN (Lemaire & Denhière, 2004)—can be tested.

Concept Selection in Working Memory

The selection of elements in working memory that occurs right after the integration phase can be made in three ways: (1) by selecting elements with an activation value over a given value, (2) by selecting the best *N* elements, *N* being a parameter, and (3) by selecting the best elements with activation values that add up to a given quantity of activation.

Episodic Memory

The functionality of episodic memory is controlled by two parameters: (1) the minimum association value for items being retrieved from episodic memory, and (2) the

```
Input? go out/fire
"go out/fire" added to working memory.
  "burn" recoverable from episodic memory but too low (.611 < .75)
  "shine" recoverable from episodic memory but too low (.311 < .75)
  "fire" recoverable from episodic memory but too low (.613 < .75)
  "burned" recoverable from episodic memory but too low (.370 < .75)
  "heat" recoverable from episodic memory but too low (.262 < .75)
  "warm" recoverable from episodic memory but too low (.546 < .75)
  "light" recoverable from episodic memory but too low (.405 < .75)
  "watch" recoverable from episodic memory but too low (.375 < .75)
  "spit" recoverable from episodic memory but too low (.617 < .75)
  "rays" recoverable from episodic memory but too low (.365 < .75)
  "notice" recoverable from episodic memory but too low (.483 < .75)
  "ax" recovered from episodic memory. Added to WM.
Looking for neighbors of go out:
  1. light (0.64) close to fire
     fire: this word is already part of the proposition
  2. went out (0.56) close to fire
  3. flames (0.56) close to fire
fire added to working memory. Looking for neighbors:
  1. flames (0.71)
  2. burn (0.69)
  3. warm (0.65)
-------------------------
Constructing the 29x29 matrix...
Integrating... (7 cycles)
Activated nodes: walk/woodcutter/forest(1.00) spit/woodcutter/fire(.867)
burn/trees(.866) wood(.830) forest(.813) go out/fire(.807) oak(.803)
woodcutter(.749) trunks(.748) drink/woodcutter/water/flask(.740)
glade(.727) branches(.703) fire(.702)
```

Table 2
Activation Values of All Words and Propositions

	Cycle			
Words	1	2	3	4
go out	.000	.000	.000	.536
go out/fire	.000	.000	.000	.726
burned	.000	.000	.000	.397
area	.288	.259	.234	.210
light	.000	.000	.000	.559
notice	.596	.536	.483	.434
notice/woodcutter/light	.693	.750	.675	.608
trees	.000	.828	.867	.863
inhale	.000	.000	.543	.488
woodcutter	.735	.806	.875	.872
drink	.000	.000	.789	.815
drink/woodcutter/water/flask	.000	.000	.860	.867
wood	.695	.847	.887	.883
drinks	.000	.000	.640	.715
burned	.000	.411	.370	.333
burns	.000	.000	.611	.778
burn	.000	.424	.381	.343
burn/trees	.000	.900	.886	.886
branches	.000	.804	.864	.864
shine	.384	.346	.311	.280
drank	.000	.000	.577	.519
oak	.781	.874	.887	.880
heat	.000	.291	.262	.236
hot	.000	.000	.679	.747
cottage	.527	.474	.681	.612
glade	.754	.854	.878	.870
spits	.000	.000	.401	.361
spit	.000	.000	.617	.555
spit/woodcutter/fire	.000	.000	.750	.870
pick	.674	.831	.878	.865
water	.000	.000	.616	.555
fire	.000	.000	.613	.796
leaves	.000	.677	.799	.719
flames	.000	.374	.733	.819
forest	.802	.882	.890	.882
flask	.000	.000	.762	.829
ax	.533	.480	.754	.809
light	.500	.450	.405	.365
luminous	.402	.362	.325	.293
objects	.317	.285	.257	.231
left	.000	.000	.562	.505
potable	.000	.000	.472	.425
potion	.000	.000	.639	.732
stroll	.599	.539	.485	.437
walk	.798	.835	.869	.857
walk/woodcutter/forest	.900	.898	.900	.900
warm	.000	.000	.546	.711
rays	.450	.405	.365	.328
watch	.463	.416	.375	.337
meet	.502	.452	.407	.366
faucet	.000	.000	.562	.506
bucket	.000	.000	.693	.758
witch	.000	.000	.548	.494
lukewarm	.000	.000	.594	.534
trunks	.000	.808	.875	.872

minimum semantic similarity with the cue word for items being retrieved from episodic memory.

User Selection, Tracing

A parameter can be set for researchers who are willing to trace the program step by step. Another parameter can be used to control the selection of neighbors by hand and to rule out possible irrelevant items. This can be used for simulating comprehension of a given text for which the researcher already knows some associated words.

CONCLUSION

This computer program is intended to help psycholinguists investigate the way in which humans comprehend texts in relation to their level of prior relevant knowledge, the situation models used, and the structure of texts processed (Baudet & Denhière, 1991; Cook & Myers, 2004; Denhière et al., 2007; McNamara, Kintsch, Songer, & Kintsch, 1996; Voss & Silfies, 1996; Zwaan & Radvansky, 1998). Researchers who are willing to explore the assets and limitations of the construction–integration model, or to compare its performance with other models—such as the landscape model (Linderholm et al., 2004) or the resonance model (O'Brien, Rizzella, Albrecht, & Halleran, 1998)—can test it on various texts quite easily.

One main interest of this program is its exhaustive model of semantic memory, which can provide associates for any word in the language. Because of the lack of such a model, previous simulations could only be run on a very limited number of texts. Researchers had to guess a few words that could be associated with all text words, resulting in small and subjective results. Kintsch (2000) and Bellissens and Denhière (2003) proposed the connection between CI and LSA; however, they did not link them in an automatic manner.

The main limitation of our model is its lack of a propositional parser that would allow free text inputs. To date, propositions have to be extracted by hand. However, the model does not need an exact description of propositions; rather, the text merely needs to be split into predicate–argument items. If the splitting is not correct, some irrelevant words could be retrieved, but they will probably be ruled out by the robust integration step. We are, however, in the process of designing a rough propositional parser, which would give us the missing link.

This program is freely available from B. Lemaire for researchers who are willing to use it for academic purposes.

REFERENCES

Baudet, S., & Denhière, G. (1991). Mental models and acquisition of knowledge from text: Representation and acquisition of functional systems. In G. Denhière & J. P. Rossi (Eds.), *Text and text processing* (Advances in Psychology, Vol. 79, pp. 155-188). Amsterdam: North-Holland.

Bellegarda, J. R. (2000). Exploiting latent semantic information in statistical language modeling. *Proceedings of the IEEE*, **88**, 1279-1296.

Bellissens, C., & Denhière, G. (2003). Retrieval from long-term working memory: A skilled use of semantic memory. *Issues in Psycholinguistics*, **2**, 145-165.

Burgess, C., & Lund, K. (1997). Modeling parsing constraints with high-dimensional context space. *Language & Cognitive Processes*, **12**, 177-210.

Caillies, S., & Denhière, G. (2001). The interaction between textual structures and prior knowledge: Hypotheses, data and simulation. *European Journal of Psychology of Education*, **16**, 17-31.

Caillies, S., Denhière, G., & Jhean-Larose, S. (1999). The intermediate effect: Interaction between prior knowledge and text structure. In H. van Oostendorp & S. Goldman (Eds.), *The construction of mental representations during reading* (pp. 151-168). Mahwah, NJ: Erlbaum.

Caillies, S., Denhière, G., & Kintsch, W. (2002). The effect of prior knowledge on understanding from text: Evidence from primed recognition. *European Journal of Cognitive Psychology*, **14**, 267-286.

Chwilla, D. J., & Kolk, H. H. J. (2002). Three-step priming in lexical decision. *Memory & Cognition*, **30**, 217-225.

Cook, A. E., & Myers, J. L. (2004). Processing discourse roles in scripted narratives: The influences of context and world knowledge. *Journal of Memory & Language*, **50**, 268-288.

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.

de la Haye, F. (2003). Normes d'associations verbales chez des enfants de 9, 10 et 11 ans et des adultes [Word association norms for 9-, 10-, and 11-year-old children and adults]. *L'Année Psychologique*, **103**, 109-130.

Denhière, G., & Lemaire, B. (2004). A computational model of children's semantic memory. In K. Forbus, D. Gentner, & T. Regier (Eds.), *Proceedings of the 26th Annual Meeting of the Cognitive Science Society* (pp. 297-302). Mahwah, NJ: Erlbaum.

Denhière, G., Lemaire, B., Bellissens, C., & Jhean-Larose, S. (2007). A semantic space for modeling children's semantic memory. In T. K. Landauer, D. S. McNamara, S. Dennis, & W. Kintsch (Eds.), *Handbook of latent semantic analysis* (pp. 143-165). Mahwah, NJ: Erlbaum.

Dumais, S. T. (2003). Data-driven approaches to information access. *Cognitive Science*, **27**, 491-524.

French, R. M., & Labiouse, C. (2002). Four problems with extracting human semantics from large text corpora. In W. D. Gray & C. D. Schunn (Eds.), *Proceedings of the Twenty-fourth Annual Conference of the Cognitive Science Society* (pp. 316-322). Mahwah, NJ: Erlbaum.

Gerrig, R. J., & McKoon, G. (1998). The readiness is all: The functionality of memory-based text processing. *Discourse Processes*, **26**, 67-86.

Glenberg, A. M., & Robertson, D. A. (2000). Symbol grounding and meaning: A comparison of high-dimensional and embodied theories of meaning. *Journal of Memory & Language*, **43**, 379-401.

Kintsch, W. (1988). The role of knowledge in discourse comprehension: A construction integration model. *Psychological Review*, **95**, 163-182.

Kintsch, W. (1998). *Comprehension: A paradigm for cognition*. Cambridge: Cambridge University Press.

Kintsch, W. (2000). Metaphor comprehension: A computational theory. *Psychonomic Bulletin & Review*, **7**, 257-266.

Kintsch, W. (2001). Predication. *Cognitive Science*, **25**, 173-202.

Kintsch, W. (2007). Meaning in context. In T. K. Landauer, D. S. McNamara, S. Dennis, & W. Kintsch (Eds.), *Handbook of latent semantic analysis* (pp. 89-105). Mahwah, NJ: Erlbaum.

Kontostathis, A., & Pottenger, W. M. (2002, December). *Detecting Patterns in the LSI Term-Term Matrix*. Workshop on the Foundation of Data Mining and Discovery, IEEE International Conference on Data Mining, Maebashi City, Japan.

Landauer, T. K. (1998). Learning and representing verbal meaning: Latent semantic analysis. *Current Directions in Psychological Science*, **7**, 161-164.

Landauer, T. K. (2002). On the computational basis of learning and cognition: Arguments from LSA. In B. H. Ross (Ed.), *The psychology of learning and motivation: Advances in research and theory* (Vol. 41, pp. 43-84). San Diego: Academic Press.

Landauer, T. K., & Dumais, S. T. (1997). A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, **104**, 211-240.

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

Lemaire, B., & Bianco, M. (2003). Contextual effects on metaphor comprehension: Experiment and simulation. In F. Detje, D. Dörner, & H. Schaub (Eds.), *Proceedings of the 5th International Conference on Cognitive Modelling (ICCM)* (pp. 153-158). Bamberg, Germany: Universitäts-Verlag.

Lemaire, B., & Denhière, G. (2004). Incremental construction of an associative network from a corpus. In K. Forbus, D. Gentner, & T. Regier (Eds.), *Proceedings of the 26th Annual Meeting of the Cognitive Science Society* (pp. 825-830). Mahwah, NJ: Erlbaum.

Linderholm, T., Virtue, S., Tzeng, Y., & van den Broek, P. (2004). Fluctuations in the availability of information during reading: Capturing cognitive processes using the landscape model. *Discourse Processes*, **37**, 165-186.

Livesay, K., & Burgess, C. (1997). Mediated priming in high-dimensional meaning space: What is "mediated" in mediated priming? In M. G. Shafto & P. Langley (Eds.), *Proceedings of the 19th Annual Meeting of the Cognitive Science Society* (pp. 436-441). Mahwah, NJ: Erlbaum.

Lowe, W., & McDonald, S. (2000). The direct route: Mediated priming in semantic space. In M. A. Gernsbacher & S. D. Derry (Eds.), *Proceedings of the 22nd Annual Meeting of the Cognitive Science Society* (pp. 675-680). Mahwah, NJ: Erlbaum.

McNamara, D. S., Kintsch, E., Songer, N. B., & Kintsch, W. (1996). Are good texts always better? Interactions of text coherence, background knowledge, and levels of understanding in learning from text. *Cognition & Instruction*, **14**, 1-43.

McNamara, D. S., & Kintsch, W. (1996). Learning from texts: Effects of prior knowledge and text coherence. *Discourse Processes*, **22**, 247-288.

Myers, J. L., & O'Brien, E. J. (1998). Accessing the discourse representation during reading. *Discourse Processes*, **26**, 131-157.

O'Brien, E. J., Rizzella, M. L., Albrecht, J. E., & Halleran, J. G. (1998). Updating a situation model: A memory-based text processing view. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, **24**, 1200-1210.

Perfetti, C. A. (1998). The limits of co-occurrence: Tools and theories in language research. *Discourse Processes*, **25**, 363-377.

Rizzella, M. L., & O'Brien, E. J. (2002). Retrieval of concepts in script-based texts and narratives: The influence of general world knowledge. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, **28**, 780-790.

Tapiero, I., & Denhière, G. (1995). Simulating recall and recognition by using Kintsch's construction-integration model. In C. A. Weaver III, S. Mannes, & C. R. Fletcher (Eds.), *Discourse comprehension: Essays in honor of Walter Kintsch* (pp. 211-232). Hillsdale, NJ: Erlbaum.

van den Broek, P., Risden, K., Fletcher, C. R., & Thurlow, R. (1996). A "landscape" view of reading: Fluctuating patterns of activation and the construction of a stable memory representation. In B. K. Britton & A. C. Graesser (Eds.), *Models of understanding text* (pp. 165-187). Mahwah, NJ: Erlbaum.

van den Broek, P., Young, M., Tzeng, Y., & Linderholm, T. (1999). The landscape model of reading: Inferences and the on-line construction of a memory representation. In R. F. Lorch, Jr. & E. J. O'Brien (Eds.), *Sources of coherence in text comprehension* (pp. 353-373). Mahwah, NJ: Erlbaum.

Voss, J. F., & Silfies, L. N. (1996). Learning from history text: The interaction of knowledge and comprehension skill with text structure. *Cognition & Instruction*, **14**, 45-68.

Zwaan, R. A., & Radvansky, G. A. (1998). Situation models in language comprehension and memory. *Psychological Bulletin*, **123**, 162-185.

NOTES

1. For the sake of readability, we are using the notions of working memory or episodic memory, but we do not claim to cover exactly the meaning of these concepts in the psycholinguistic literature. Because of computational requirements, these notions are simplified in comparison with their theoretical counterparts.

2. *Oiseau* (*bird*) is not considered, because it is already part of the proposition.

(Manuscript received May 4, 2005;
revision accepted for publication August 31, 2005.)