Book Reviews

Time-constrained Memory: A Reader-based Approach to Text Comprehension

Jean-Pierre Corriveau
(Carleton University)

Mahwah, NJ: Lawrence Erlbaum
Associates, 1995, xx+408 pp;
hardbound, ISBN 0-8058-1711-5, $79.95;
paperbound, ISBN 0-8058-1712-3, $34.50

Reviewed by
Arthur C. Graesser
The University of Memphis

The computational model proposed in this book is incompatible with any theory that assumes that a single, invariant interpretation is built as a result of comprehending a particular text. The book also challenges the claim that traditional formal theories of syntax, semantics, and reasoning can adequately implement text comprehension on a computer. Corriveau's computational model is called IDIoT, which stands for "Idiosyncratically-Directed Interpretation of Text." According to IDIoT, the interpretation of a text depends on the individual reader's knowledge and mental state at a particular point in time. Therefore, the interpretation is generated by a mechanism that is nondeterministic (i.e., it varies from person to person) and diachronic (i.e., it varies across time).

The major assumptions of IDIoT are embraced by most contemporary models of comprehension in cognitive psychology and discourse processing (Britton and Graesser 1996; Weaver, Mannes, and Fletcher 1995). In particular, most psychological models assume that memory plays a critical role in constructing interpretations. Long-term memory is a vast repository of knowledge units that get activated during comprehension, in a limited-capacity working memory (and short-term memory). Text interpretation fluctuates among readers to the extent that readers have different knowledge units in long-term memory and different spans of working memory. According to IDIoT, the activation and processing of the knowledge units interact in parallel, much in the spirit of Minsky's *Society of Mind* (1986) and connectionist models. It takes a nontrivial amount of time for some knowledge units to be activated and to complete their processing steps; if the processing is not completed by some deadline temporal duration, then a knowledge unit might not have any impact on the final interpretation of a text. Readers differ in their processing time parameters (such as learning rate, activation rate, and memory decay rate), which also results in fluctuations in interpretations among readers. Once again, these basic claims about memory and processing time are adopted by many of today's researchers who develop psychological models of reading and comprehension, so these researchers would applaud Corriveau's efforts in the field of computational linguistics.

Corriveau uses IDIoT to simulate a broad spectrum of linguistic and discourse phenomena: constructing syntactic trees, resolving the referents of nouns and pronouns, determining the context-appropriate sense of an ambiguous lexical item, and generating knowledge-based bridging inferences. When IDIoT receives a sentence or a short excerpt of two or three sentences (long texts are beyond its present capabilities), a human user must declare what knowledge units it knows about and the values of various parameters that refer to capacity and processing time. One or more interpretations of the input text are produced by IDIoT, along with a trace of the processing steps at different levels of linguistic and discourse analysis.

Unfortunately, it was sometimes difficult to determine where to assign credit or blame when examining the model's output. When IDIoT produced interesting output, was the success due to the knowledge units declared by the user, to particular processing time parameters, or to components of IDIoT's computational architecture? This question could be answered by performing a systematic evaluation of the model and a sensitivity analysis on the importance of various components. However, it was beyond the scope of Corriveau's book to systematically evaluate the success of IDIoT. He never reported data on the extent to which his model could handle a corpus of naturalistic texts (on relevant evaluation criteria), nor did he fit the model to available psychological data at a fine-grained level.

Some critics will fault Corriveau for skirting attempts to systematically evaluate his model, whereas others will allow him some leeway in his early stages of model development. In either case, the book is more of a "proof of concept" than evidence of a decisive, successful model that has survived some critical tests.

This book makes two serious contributions to the scientific study of text comprehension. Regarding the first contribution, Corriveau covers theories of comprehension in diverse fields, including computational linguistics, artificial intelligence, cognitive psychology, cognitive science, and discourse processes. Whereas most researchers stay encapsulated within their pet fields, Corriveau has the courage to adopt a more interdisciplinary stance. I was genuinely impressed with the scope of the work cited in this book. Corriveau also critically evaluated the theories and foundational assumptions in the various fields, but I was not uniformly impressed with the accuracy or depth of these critiques. Regarding the second contribution, Corriveau has presented a detailed computational model that specifies the representations and processes of readers during text comprehension. The existing models in psychology have not implemented all of the mechanisms of text comprehension on computer, whereas existing models in computational linguistics and artificial intelligence are many steps removed from psychological data and constraints. This book helps narrow the gap between cognitive psychology and computation. As a next step, I encourage Corriveau to team up with an accomplished researcher in cognitive psychology and discourse processing.

References

Britton, Bruce K. and Arthur C. Graesser, editors. 1996. *Models of Understanding Text.* Mahwah, NJ: Lawrence Erlbaum Associates.

Minsky, Marvin. 1986. *The Society of Mind.* New York: Simon and Schuster.

Weaver, Charles A., Suzanne Mannes, and Charles R. Fletcher, editors. 1995. *Discourse Comprehension: Essays in Honor of Walter Kintsch.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Arthur C. Graesser is a professor in the Departments of Psychology and Mathematical Sciences at The University of Memphis. His research investigates cognitive models of comprehension, knowledge representation, connected discourse, inference generation, and question answering. He is an associate editor of the journal *Discourse Processes.* Graesser's address is: Department of Psychology, The University of Memphis, Memphis, TN 38152; e-mail: a-graesser@memphis.edu

The Trouble with Computers: Usefulness, Usability, and Productivity

Thomas K. Landauer
(University of Colorado)

Cambridge, MA: The MIT Press, 1995,
xiii+423 pp; hardbound, ISBN
0-262-12186-7, $27.50

Reviewed by
Harold Thimbleby
Middlesex University

The first computer was developed by Charles Babbage. It cost more than a locomotive and did not work properly. Even though everyone tells us that computers are now "more powerful," the basic story does not seem to have changed. Computer projects are still costly and ineffective. Over the last few decades, industry has invested heavily in computers, yet has shown flat, if not decreasing, productivity. Spreadsheets, inventory control, management information systems—weren't they designed specifically for industrial productivity?

You use computers. Are your programs as effective as you wished? Has your research productivity gone up—even when you take account of the time it takes to get the programs working, and when you take into account the lost data when you hit 'OK' by mistake? And would you like to know how to make your use of computers more effective?

When a paper is published in *Computational Linguistics* that relies on a computer, do you believe it? If you think the authors have got a point, which you would like to test, extend, or develop, can you do so easily by working with the same programs as the original authors did? I have written to authors of papers (in other journals) and asked if they could allow me to use their programs; I've often had replies that are tantamount to admitting the programs do not exist in *quite* the way they were written up in the published papers. Further, if you can get hold of and use the original programs, do you have ways to check that they are valid? To what extent, then, do computers contribute to the actual progress of computational linguistics rather than to its imagined progress?

If the so-called performance enhancements that computers have achieved in the last decade had been achieved in almost any other technology, such as land transport, then the world would certainly have been transformed beyond recognition. Yet computers have done no such thing. We may be bombarded with facts about how fast they are, but they do not seem to get any more work done for us. Every year they will get faster, as they have been doing since they were invented. Every year people will buy more of them. Yet somehow the world will not become a better place (except perhaps for the consultants and trainers).

Landauer's discussion in the first part of his book, and his analysis of all sides of the arguments—for and against—is one of the most thorough and well written that I have seen. He has certainly given me more ammunition for the battle of getting computers to be used thoughtfully. Elsewhere there is remarkably little computer-critical thought around. Take the *Scientific American*, which, as Landauer says, goes all wild-eyed when dreaming about the future of computers in a way that it would not with medicine or any other subject. Somehow the fantasies of the computerphiles, of utopia-just-around-the-corner (when computers become *just a bit* faster, or *just a bit* smaller), infect even a usually rigorous, scientific journal.

Computers are hard to use effectively, and the main reason is that they allow people to make gross errors that are hard to fix. A whole day's work can be zapped with a misplaced mouse click. There are few other aids to modern life that are so perverse!

A more quantified way of showing the scope for low productivity is the following comparison: If you ask twenty expert typists to type a letter, the slowest takes about 30% longer than the fastest. Put a group of twenty word-processing typists onto word processors, and the slowest takes *six* times as long as the fastest. In programming, the comparisons are extreme: the slowest programmer (from a sample of twenty) takes fifty times longer than the fastest. In other areas of human effort, say athletics, ratios of performance across the same-sized groups of twenty subjects is about two. Apart from anything else, the data suggest that if you select computer users at random, productivity will be much worse than optimal.

Similar cynical thoughts have been expressed elsewhere: Clifford Stoll's *Silicon Snake Oil* (1995), which is technological, and Stephen Talbott's *The Future Does Not Compute* (1995), which is oddball and nonscientific (and hence argues some unfamiliar points). Where Thomas Landauer shines through is, first, his discussion is more balanced and academically sound, backed by empirical data and evaluation; second, he tries to explain why; and, last, he promotes a recipe to solve the trouble.

Landauer's explanation is disappointing—especially after the careful fairness of his arguments, after his recruitment of a wide-ranging literature—but close enough to the truth: he says we love computers, so we use them even if they are not strictly economically sound investments. I think there is more to it than that. Computers are fashion accessories, and there is more to be said about exploitation by subtle marketing: but if it boils down to love (and greed), which are hardly scientific or technical concepts, how are we to plan and progress?

Landauer's recipe for improving system performance is the most important part of his book. He has persuaded us that computers are a mess, and he has done so carefully in a way that would persuade the most unrepentant technophiles. He has tried to explain why we put up with computers. Now the rest of the book turns to ways of avoiding low productivity and disasters.

It is well known that designers often design for themselves unless they have been trained to realize that people are diverse, and that users are unlikely to be like them. In computing, the design problem is exacerbated because designers, who are self-selected to be computer competent, are typically many times better at using computers than everyone else. In contrast, in any other common area of design—say, designing chairs—the designers would be unlikely to have such special and distinctive skills that marked them out as so different from everyone else. Who is so good at, say, sitting that they might underestimate how hard everyone else finds using chairs? But what is ridiculous in chairs is commonplace in computing. It is likely that researchers in computational linguistics, to take just one speciality allied to computing, are not only better at these subjects than most other people but are very much better. So to the extent that we build programs for others to use—even to advance our research—we are probably not getting the best out of them because the users are so different from us.

User-centered design is Landauer's way of improving productivity. As a general-purpose acronym, UCD, Landauer means user-centered **design**, user-centered **development**, and user-centered **deployment**. UCD is a way of solving human problems, rather than concentrating too exclusively on technical problems—which the first half of the book has shown to be surprisingly counter-productive. UCD concentrates on getting empirical data on how systems help (or hinder) people working with computers. He shows that the cost–benefit of UCD is very impressive: it is cheap and it is effective.

The Trouble with Computers finishes with a personal view of "fantasy business systems," future systems designed, developed, and deployed in this way. By his own admission, fantasizing is not enough, when you've just persuaded your readers to collect empirical data! I did not find the examples impressive. In fact, they made it more obvious that Landauer has forgotten theory.

The reason why it is so important to collect empirical data about how people use computers and why recruiting these data to the design process has such dramatic effects is, I suggest, because computer system design is typically a-theoretical, and the people working with computers—despite Landauer's claims about their flair—are relatively incompetent and unimaginative with respect to the complex computer design task they undertake.

Consider this analogy: If buildings were built by decorators with no idea beyond wall paint, no ideas about structural engineering, and no idea that one might subcontract, then empirical data about how buildings fail would help improve building utility enormously. One would also be tempted, as UCD suggests, to build prototypes, test them, and involve users in testing to help find the problems the so-called designers missed. It would surely be easy to show how cost-effective UCD would be in this imaginary world of bad design.

In safety-critical areas, such as medical applications, adequate testing may be impractical: the designs have to work well enough before they are delivered. Lengthy testing might not exercise enough of a design to provide any significant data. And that means systems have to be *known* to work well enough a priori—that is, by their design being based in theory. Users are not going to be able to explore much of a design or contribute much to the generality of the language-processing algorithms. UCD, if you still want to call it that, has to be done in theory or in simulation by the designers, not by using empirical data as a substitute for thinking.

UCD will still be necessary when programmers are more competent (or when better theories are available to aid design); but in the meantime, programmers' competence probably has a greater influence on why systems fail than any other single factor.

The Trouble with Computers is a challenge to everyone who works with computers: first, that we can do better—embarrassed into action by Landauer's exposé of computer failure and under-achievement; secondly, that by developing theory we, as active scientists developing computer systems and theories, can do better than users can either express or expect.

To summarize: *The Trouble with Computers* is a good book. It should be read by everyone who uses or buys computers. For people who design computer systems, or who work in computational theories it is, more so, a challenge to do even better, for I am sure we have a lot of catching up to do. If Thomas Landauer is right when he would have us believe that we use computers because we love them, it is time we made them worthy of our attention.

References

Stoll, Clifford. 1995. *Silicon Snake Oil: Second Thoughts on the Information Highway.* Doubleday.

Talbott, Stephen L. 1995. *The Future Does Not Compute: Transcending the Machines in Our Midst.* O'Reilly & Associates.

Harold Thimbleby is Professor of Computing Research and Faculty of Technology Director of Research at Middlesex University, London. He has over 200 publications, centering upon human-computer interaction. Thimbleby's address is: Computing Science, Middlesex University, Bounds Green Road, London, England N11 2NQ; e-mail: harold@mdx.ac.uk; URL: http://www.cs.mdx.ac.uk/harold