20 How do computational models help us develop better theories?

Dennis Norris
MRC Cognition and Brain Sciences Unit, Cambridge, UK.

What's a theory?

Computational models are a good thing, or so most of us believe. But, what are they actually good for, and how do they help us develop better theories? Here I assume that even if there is somebody out there who doesn't think that models are a good thing, we all think that theories are fundamental to any scientific enterprise. Although consideration of the nature of scientific theories is a big enough topic to occupy the entire academic career of a sizeable proportion of the planet's philosophers, I will focus on what I consider to be the central component of a scientific theory: To provide an explanation of some phenomena, observation, or set of data. That is, to explain how and why something happens.

Is a computational model a theory?

It is easy to fall into the trap of thinking that computational models are just theories. But computational models don't necessarily provide the essential ingredient of a theory: They don't necessarily explain things. For example, in itself, a mathematical model or computer simulation might not explain anything, even if it does a great job at simulating the data. Ken Forster has illustrated this by asking us to imagine that his next door neighbour can make the correct prediction about the outcome of any experiment on word recognition (Forster, 1994). Does that make Ken's next door neighbour a good theory of word recognition? Clearly not. The existence of Ken's neighbour doesn't explain how or why human

word recognition works the way it does. On the other hand, I'd love to have access to the proclamations of Ken's imaginary neighbour. It would save the tedious task of running experiments. The important message of Forster's illustration is that a model needs to amount to something more than replacing one thing we don't understand with something else we don't understand. A computational model is of no value unless we can understand what it does. This issue was a real concern in the early days of connectionist modelling (cf McCloskey, 1991). Some might argue that it continues to be a problem with much current connectionist modelling. If a connectionist network is trained to perform some task, such as reading aloud (e.g., Seidenberg and McClelland, 1989) and, after training, produces human-like behaviour, how much does this actually explain? In itself, the fact that a model can simulate the data doesn't tell us anything other than that a model of this sort can perform the task. Until we understand how it works, such a model provides no more explanation than Ken's neighbour. But, at least Ken's neighbour can predict the outcome of experiments! The real insights from connectionist models generally come from examining the networks to discover how they manage to perform the task, and by comparing the performance of networks with different architectures or representations. In themselves, computational models are no substitute for good theories.

On the other hand, we shouldn't dismiss models entirely just because we don't understand how they work. Models are often valuable because they can provide proof that systems embodying particular computational principles or mechanisms really are capable of simulating the data. That is, they provide proof of principle. If someone claims that the data is inconsistent with models with property X, and it is then shown that a model with property X can simulate the data, then the simulation makes an important theoretical contribution, even if no one is quite sure how it works.

Some people hold that computational models are just formalisations of theories. This is slightly different from the view that models are the same as theories, because it implies that you are starting from something that offers some kind of explanation. For example, Herb Simon (2002) writes that "The term 'computational model' is used for a theory that is stated definitively enough that precise reasoning... can be performed on it to infer the course of system behaviour." While this might occasionally be the case for some theories, and perhaps is often the case for mathematical models in disciplines such as physics, it is rarely true of computational models in psychology. That is, there is rarely a straightforward one-to-one mapping between model and theory. As I will illustrate later, when one tries to construct a computational model of

a psychological theory, the model often contains assumptions not in the theory. For example, a theory might be neutral with respect to how some particular process is implemented. But some decisions or assumptions need to be made to make the simulation work. Also, for good reasons, models often omit important parts of the theory. For example, some parts of a theory might be just too hard to simulate, but one wouldn't want that to rule out any computational simulations of the theory. Parts of the model might be simplified, or fed with hand crafted input, so that other parts of the theory can be tested. For example, models of written or spoken word recognition rarely concern themselves with the early stages of visual or auditory perception. These processes are generally outside the scope of the models, but the models clearly need to be supplied with some input. Maybe such models just wouldn't qualify as 'computational models' according to Herb Simon, but they can be very useful.

WHY BUILD COMPUTATIONAL MODELS?

If computational models aren't really theories, then why build them? Well, computational models might not be isomorphic with theories, but they can be used to formalise the process of generating predictions from theoretical assumptions. That is, they provide a crucial link between theory and data. The psychological literature is full of debates that read like pantomime dialogue: "My theory predicts that", "Oh no it doesn't", "Oh yes it does". A properly formulated computational model can help resolve these debates by providing a formal demonstration that, when implemented as a computer program, the theory really does (or doesn't) make the predictions claimed of it. Even setting aside the technical difficulty of constructing simulations, this isn't a trivial task. As we will see later, one must be absolutely certain that the final behaviour of the model is driven solely by the critical theoretical assumptions under investigation, and not by things that are either added to, or left out, of the simulation.

Models of word recognition provide a good example of how theoretical progress is critically dependent on the development of properly formulated computational models. In research on word recognition, models don't just resolve debates over what theories predict, they are often the only way that even the theorists themselves can be sure what their theories predict. It is no surprise that almost all influential models of spoken or written word recognition are computational. The behaviour of these models is always dependent on the actual set of words that are in the model's lexicon, and how the representations of these words relate to each other (e.g., effects of neighbourhood density or

competition). That is, the behaviour of the models doesn't follow directly or predictably from the mechanisms and processes in the model. The behaviour is determined by the interaction between those mechanisms and the words in the lexicon. Different words will be processed differently depending on which other words are in the lexicon. Trying to work out in your head how thousands of words interact is an impossible task for mere mortals. Simulations are essential. For example, in some of our own work on the Shortlist model, it has proved difficult to even hazard a guess as to how certain changes will alter the model's performance.

Another area where the behaviour of models is hard to predict is the case of stochastic models. Many models (e.g., Page and Norris, 1998; Ratcliff, 1978) assume that some components of psychological processes are prone to noise - that is, the outcome of the processes is not always the same. Frequently there is no simple analytical procedure (i.e. no mathematical formula) that can predict the average outcome of these processes, and the only option is to run the model many thousands of times. The models produce noisy data, just like real human subjects do, and the outcome is impossible to predict with any precision. The frustrating thing here is that even the behaviour of very simple models with very few parameters becomes hard to predict once noise is added.

This is why computational models have become indispensable in many areas of cognitive psychology - we just couldn't make the link between theory and data without them.

Why you should build a model even if you don't think you need one

Building models is a valuable theoretical exercise - even if they don't work. Building models makes you think. When you sit down and try to cast your theory as a computer program, sometimes that bright idea you had turns out not to be so clever after all. Modelling make you think about the problem at a level of detail well beyond what most of us apply to our verbal theories. With verbal theories, it is very easy to skate over hidden assumptions and convince yourself that your wonderful new creation really does explain the data. When you try to formulate your ideas as computer code, you frequently realise that there is something missing from your theory, or that a particular mechanism you propose just won't behave the way you thought it would. So, it's back to the drawing board to see if you can rescue your precious theory, or maybe you even have to concede that it's never going to work at all. You can make all of this valuable progress before you've even written a line of code!

It is sometimes said that the hallmark of a mature science is cumulative progress. A science comes of age when there is a sound theoretical bedrock on which we can build ever more detailed and refined theories. As theories become more complex, the formal discipline of modelling is almost the only way to achieve that cumulative progress. Much verbal theorising is fragmented. Each report of new data is accompanied by a new theory that sometimes doesn't apply to much other than the new data itself. If a verbal theory is extended, there is always a worry that the new parts of the theory might have implications for the way the theory accounts for existing data. When you extend a computational model to explain new phenomena, or to accommodate new data, it's relatively easy to check that nothing you've added to the old model has changed it's predictions. In other words, you are forced to develop an integrated theory that explains new phenomena by extending the scope of the previous theory. This is sometimes referred to as "nested theory development". I will say little more about this here because it is the focus of the chapter by Ardi Roelofs.

A DOUBLE DISSOCIATION BETWEEN MODELS AND THEORIES

I've tried to make the case that models and theories aren't the same thing. I'll illustrate this by the classic neuropsychological technique of demonstrating a double dissociation between the two. There are theories that don't have models, and models that don't have theories. Both theories with no models, and models with no theories, fall into the "Oh yes it does", "Oh no it doesn't" category. That is, it is hard to say that these theories and models are right or wrong. But that is exactly what is wrong with them. I'll illustrate this dissociation with a couple of my least favourite examples. The choice of examples is driven entirely by personal prejudice.

Short-term memory is simply activation of long-term memory

Some memory researchers have suggested that there is no independent short-term memory (STM). STM is nothing more than the temporary activation of corresponding representations in long-term memory (LTM) (e.g., Ruchkin, Grafman, Cameron and Berndt, in press). This suggestion might seem attractive because of its parsimony. Why postulate two stores if you can do it all with one? The usual counter-argument to this view is to cite neuropsychological evidence for a dissociation between LTM and

STM (e.g Vallar and Baddeley, 1984). But this argument doesn't seem to carry much weight with STM-as-LTM theorists. It is worth noting that while there are several computational models of STM (Brown, Preece and Hulme, 2000; Burgess and Hitch, 1992,1999; Henson, 1998; Page and Norris, 1998), there are no models that treat STM as activation of LTM. Maybe there is a good reason for this. One of the fundamental problems that face any model of STM is how to remember lists like "1,1,2,1,1" in the correct order. That's certainly not difficult for people. But, it requires representing that the same digit, which we would expect to have a single long-term representation, is repeated 4 times. This is somethimes referred to as a *binding* problem. If you have a STM that can contain representations that are copies of LTM representations, or are bound to LTM representations (e.g., with the equivalent of pointers from STM objects to LTM) it isn't difficult to store such sequences. If the only mechanism you have is LTM activation, then this sequence is likely to be remembered as simply "1,2". That is, the LTM representation for "1" will be strongly activated, and the representation for "2" will be weakly activated. The LTM representations won't code the number of digits, or the relative position of the "1"s relative to "2". It is simple to say that STM is activated LTM, but not so simple to produce a computational model demonstrating how this might work. It is probably even harder to show that this could be done without introducing some specialised mechanism that was responsible solely for the storage of short-term information i.e. a STM. So, here is a theory with no model. In fact, because there is not even a hint of how it might work in practice, it probably isn't a theory at all. That's the moral of this short story: If you have no idea how your theory might be translated into a model, it probably isn't a very good theory in the first place. I can't resist adding another example here. Along with James McQueen and Anne Cutler, I have argued that there is no benefit to be had during word recognition in having information from the lexicon feed back to influence the identification of pre-lexical units such as phonemes (Norris, McQueen and Cutler, 2000a). As the commentaries on Norris et al (2000a) reveal, not everybody is convinced by our argument. But we are still waiting for those who believe that feedback is beneficial, to produce a model to support their claim. More importantly, we're still waiting for them to produce a theory.

Models with no theories

The idea that you could develop a model with no theory might seem rather strange. If you have a model, then surely you must have had a theory? Surprisingly enough, there are some models that simulate data

very precisely, but fail to explain how human behaviour produces the data simulated by the model. An extreme case might be a model that could be tweaked to simulate any conceivable pattern of data. Such a model wouldn't tell us anything worthwhile at all (cf Roberts and Pashler, 2000, 2002). The theory would be "Anything can happen". Many early connectionist networks were also 'models with no theories', because the theory was often little more than "Things are done with distributed representations" (cf Forster, 1994, for a critique of such models). However, in some cases, it is less immediately obvious that the model doesn't really offer an explanation. The simulations look good, and the model does incorporate some sensible theoretical principles.

Two example of models without a proper theory are the perturbation model of memory (Lee and Estes, 1977,1981) and simulations of eye-movements during spoken word recognition presented by Allopenna, Magnuson and Tanenhaus (1998). The perturbation model simulates serial position curves, mainly in immediate serial recall experiments. In these experiments subjects are presented with a sequence of items (frequently digits or letters) and are required to recall the items in the correct order. The perturbation theory is a statistical model that generates predictions of the probability that each item in a list is recalled at each output position. For example, it simulates the fact that items at either end of a list are more likely to be recalled in their correct position than are items in the middle of a list. The problem is that the model doesn't explain how any particular list is actually recalled. In effect, the model describes the data, but doesn't describe the behaviour or mechanism that produces the data.

Allopenna et al present simulations using TRACE (McClelland and Elman, 1986) that suffer from exactly the same problem. They simulated data from an experiment tracking listeners' eye movements to pictures of objects as subjects listened to sentences. The sentences could contain either the names of those objects, or the names of phonologically similar objects. For example, the sentence might be "Pick up the beaker, now put it in front of the diamond" and there might be pictures of a *beaker*, a *beetle*, a *speaker* and a control such as a *carriage*. Activation of the spoken words was simulated in TRACE. TRACE can simulate the probability of recognising each word at each moment in time. Plots of these probabilities show an excellent fit to the eye movement data (i.e. the probability of fixating on the different pictures). The fit is so good, that surely it must be a good model (see Roberts and Pashler, 2000, 2002, for further discussion of the relation between good fits and the quality of models). But, consider what subjects must be doing whenever the model predicts that they are equally likely to fixate either of two pictures.

Clearly subjects can only be fixating one of the two pictures at any point, but the model doesn't say which one. In fact, the model doesn't say how long each picture will be fixated for either. Predictions at one point in time are completely independent of the following point in time. The model therefore predicts that, from moment to moment, the subjects' eyes flicker randomly from picture to picture. Although the model does a very good job at simulating behaviour averaged over trials, it doesn't give a plausible explanation for what subjects do on an individual trial. To use terminology that Tanenhaus and Magnuson use elsewhere, the model doesn't have a linking hypothesis (Tanenhaus, Magnuson, McMurray and Aslin, 2000).

SHORTLIST: A CASE STUDY OF THE RELATIONSHIP BETWEEN MODEL AND THEORY

Shortlist (Norris, 1994) is a model of how we recognise the words in continuous speech. The overall structure of the computational model is shown in Figure 20.1. The input to the model is a sequence of phonemes. A lexical lookup process then identifies all of the words that corresponds to sequences of phonemes in the input. So, for example, the input "catalogue" would match the words "cat", "cattle", "a", "log", and "catalogue". These words are then entered into an interactive activation network where words that overlap in the input (e.g., 'cattle' and 'log') are connected together by inhibitory links. Words in the network receive input proportional to the number of phonemes they contain. Because words that overlap in the input inhibit each other, they are unlikely to be activated together. The network therefore settles on a set of activated words that don't overlap each other, and this almost always corresponds to the sequence of words intended by the speaker.

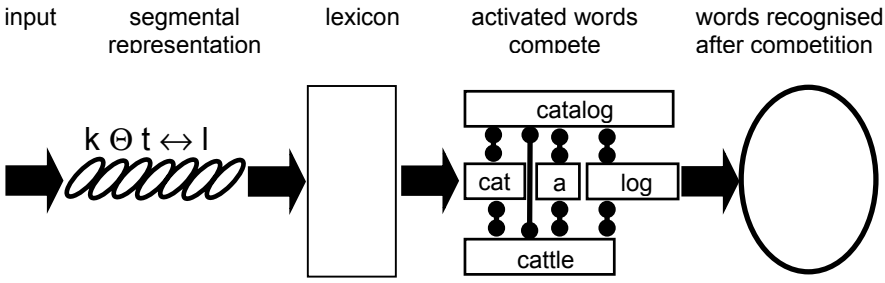

FIG. 20.1. Shortlist.

Fundamentally, the model is very similar to TRACE (McClelland and Elman, 1986). The major difference is that whereas TRACE has feedback from lexical to pre-lexical representations, Shortlist has a completely feedforward architecture. Furthermore, TRACE has a multiply duplicated lexicon so that there is a node corresponding to every word in the lexicon starting at every possible phoneme position in the input. In Shortlist there is a single lexicon and words are only considered in the interactive activation process once they have bottom-up support from the input.

The central theoretical assumptions behind the original Shortlist model are listed below:

T1 The flow of information from pre-lexical to lexical levels is bottom-up only. This was the central motivation for the original Shortlist model.
T2 Bottom-up selection of multiple lexical candidates is based on both matching and mismatching information. (i.e. a claim about the procedure for computing a match between input and lexical entries).
T3 Matching lexical candidates (and only those candidates) enter into a competition process that optimises the parsing of the input into words.
T4 There is no need for explicit lexical segmentation. (i.e. the model doesn't need to be told where words begin and end in the input).

Now the assumptions in the computational implementation of Shortlist:

M1 The input to the model is a string of phonemes
M2 The input contains no phoneme deletions, insertions or substitutions (i.e. there are no errors in the perceptual analysis).
M3 The dictionary contains a single canonical representation of each word. (i.e. no account of pronunciation variation).
M4 Lexical lookup is by means of a serial search through a dictionary.
M5 The match between the input and lexical entries is computed by counting +1 for each matching phoneme in the correct position, -3 for each mismatching phoneme.
M6 Matches between input phonemes and the corresponding phonemes in a lexical entry are all-or-none (i.e. there is no account of phoneme similarity).
M7 The candidates are entered into the network just by wiring them in as required.
M8 Overlapping candidates are connected by inhibitory links.
M9 Competition is performed by an interactive-activation network.
M10 The model output is a pattern of lexical activations over time.

None of these assumptions is an assumption of the underlying theory. To make the model work, I had to make at least 10 extra assumptions that aren't stated in the theory. Some of these assumptions concern issues where there seem to be no overwhelming theoretical reasons to constrain the way the model works. For example, the model uses a string of phonemes as input, but the model would work just as well with a string of phonetic features. Discovering the exact form of pre-lexical representations is the "big" question in speech recognition. It would be crazy to decide that we shouldn't build models of word recognition until we have first solved this problem.

The other main class of assumptions are those where simplifications have been made to make the model work. For example, we could extend the model to deal with cases where the input might be misanalysed so that the phonemes presented as input to the model might have some phonemes added, or some deleted. But this didn't seem worth doing, as it wouldn't help in simulating any of the data sets that Shortlist has been applied to.

In Norris (1994) it was very much left as an exercise for the reader to work out which assumptions were part of the "core theory" and which were just pragmatic moves to implement the model. This has led to some confusion as to what the underlying theory really is. The decision to perform lexical competition with an interactive-activation network with inhibitory links between competing words has sometimes been taken to be a core theoretical claim. [Indeed, at least one member of this workshop thought it was]. However, hiding away in the original paper is a note that the same function could have been performed by the kind of dynamic programming techniques used in automatic speech recognition. It isn't necessary to understand how dynamic programming works. The important message here is that it is very different from an interactive-activation network, but it should be possible to make a dynamic programming algorithm behave just like the network - and it is (Scharenborg,, Norris, ten Bosch and McQueen, in prep). This illustrates that we often know that there are alternative ways in which a model might compute some function, but we have no data, or a priori reason, to choose one of these ways rather than another. In effect, we are saying that the theory is neutral as to the exact implementational details. However, the theory does need to be implemented, so we have to make a choice. But we need to remember that this is a modelling assumption, and not a core part of the theory. Of course, that doesn't mean we aren't interested in how competition really happens. It means we haven't got a clue how it really works, and we are hoping that the pragmatic solution we have adopted behaves much like the real thing. If later on we find that

competition is computed in a completely different way, and that it actually behaves differently, then we could be in trouble.

One way of understanding competition in Shortlist is in terms of Marr's (1982) distinction between computational, algorithmic, and implementational levels of description. The whole idea of competition is part of a computational analysis of the problem. If there are no reliable cues in the signal to indicate where words begin and end, there just has to be a mechanism like competition. This is true irrespective of whether we are talking about human speech recognition, or automatic speech recognition systems. The need for competition follows from an analysis of the task of recognising connected speech - it is the only way to do it. But Shortlist is neutral with respect to the choice of algorithm or implementation. How competition is performed involves a claim about algorithms. One might want to claim that competition really does involve inhibition of the sort that is embodied in an interactive activation network, and not some form of dynamic programming. These are two alternative algorithms that achieve the same computational result. Currently there seems to be no principle or data that would allow us to choose one algorithm over another. Of course, eventually one would hope that it would be possible to make informed claims about how algorithms are actually implemented in the brain, but that is something we are unlikely to be able to do in the near future for higher level cognitive processes,

One further part of the model that warrants comment is the process of incorporating candidate words in the shortlist into the competition process. This is another example of the binding problem mentioned in the discussion of STM. There I drew attention to the problem of memorising a string containing more than one token of a particular digit. Intuitively at least, it seem implausible that there are multiple identical lexical representations of digits, or any other words. If that is the case, how do we construct a representation that might need to contain several tokens of the same word? We (Norris, McQueen and Cutler, 2000b) titled one of our papers "Feedback on feedback on feedback: It's feedforward". Maybe this is hard to understand out of context, but it would be even harder to understand if you could only represent one token of the word "feedback". Representing multiple tokens isn't too difficult in the STM case, because the representations of items in a list can be separate. But, in speech, potential candidates can overlap in the input. Whether a word is recognised or not depends on how it overlaps with other words. It isn't too difficult to write a computer program to do this, but it isn't at all clear how the brain might perform such a task. Working out how the brain solves the binding problem is another one of those "big" questions.

What did we gain from building the Shortlist model?

First, it helped make the case that the lexical-feedback used in TRACE, wasn't actually necessary. Autonomous feed-forward models work fine. Second, TRACE effectively treats the entire lexicon as candidates. Shortlist demonstrated that you need very few candidates in the candidate set. Both of these can be considered to be "proof of principle" demonstrations. The model shows that particular computational principles can perform particular functions. In other words, the mechanisms proposed by the theory really will do what they are supposed to do. Having established that the underlying principles were sound, we were then able to try to make the critical "cumulative progress". We performed experiments to show that there really was the kind of competition process proposed in the model (McQueen, Norris and Cutler, 1994; Norris, McQueen and Cutler, 1995), which enabled us to reject the early version of the Cohort model (Marslen-Wilson, 1987; Marslen-Wilson, and Welsh, 1978).

Second, having established the viability of a bottom-up model we could build on that and try to develop the model further. The model has undergone three significant revisions (Norris et al, 1995, 1997, 2000a). Interestingly, the most substantial revision was motivated by the discovery that the model did something that human listeners never did. In both McQueen et al (1994) and Norris et al (1995) we asked listeners to press a button when they heard a real word embedded at the start of a nonsense word. One of the filler stimuli was the nonsense word "jump↔v". Shortlist identified "jumper", which was never noticed by the listeners, nor indeed, by the experimenters who designed the materials (in British English, the final 'r' in 'jumper' is not pronounced).

Did this departure from human performance reflect a flaw in the theory? Well, it certainly was a deficiency in the theory, but it was a huge plus for the model. The model had drawn our attention to something we had never even thought about. It seems that people find it very hard to spot words in this task when this would leave a single consonant or consonant cluster remaining. In English, content words always contain vowels. Listeners behave as though they are trying to parse the nonsense words into a sequence of "possible words". The phoneme [v] has no vowel and is not a possible word, and this makes listeners less likely to accept the segmentation "jumper" - "v", than "jump" "↔v".

After confirming this informal observation experimentally, we went on to extend the model to develop what we referred to as the Possible Word Constraint (Norris, McQueen and Cutler and Butterfield, 1997). An added bonus of this modification of the model was that it provided a better and simpler account of earlier data on speech segmentation that

we had simulated in the model (Norris et al 1995). A failure of the model had led us to develop a simpler model that accounted for more data. A rare occurrence indeed, but one that would never have happened without a computational model.

WHAT DOES IT ALL MEAN?

Let's suppose that you have developed a model, it simulates the data, and you know how the model works. This is the point where most modellers just bathe in that warm glow of self-satisfaction. Make the most of it, it only lasts until the reviews of your paper come back telling you that you should add a few extra simulations of the reviewers favourite bits of data. But is that the end of the story? No. Now comes the bit that most modellers don't even bother with. Getting a model to work is only one step toward answering the "how" and "why" questions that lead to a real explanation. For any set of data, there is a potentially infinite set of theories (almost all of them wholly implausible) that would be consistent with the data. The same is true of models. So, what is it about your model that makes it able to simulate the data?

I've already touched on this issue when I tried to separate out the model-specific assumptions from the theoretical assumptions in Shortlist. What I tried to do was to abstract away from the implementation, and concentrate on the set of theoretical principles that were really explaining why the model works. In Shortlist, lexical competition is important, but the way it is implemented probably isn't. The explanation lies in the competition, not in the fact that it uses inhibitory connections between words. The model works because it has particular properties, and one of them is competition. One could build other models that might appear very different but, if they had the same properties, they could still simulate the data. What we need to know about models is not just that they work, or even how they work, but why they work.

This is clearly expressed in the following quotation from Grossberg (1987):

> "As each research group injects a stream of new models into this sprawling literature, it becomes ever more essential to penetrate behind the many ephemeral differences between models to the deeper architectural level on which a formal model lives. What are the key issues, principles, mechanisms, and data that may be used to distinguish one model from another? How may we decide whether two seemingly different models are really formally equivalent, or

> are probing profoundly different aspects of cognitive processing. (pp. 23-24)"

This is an issue that is especially important for understanding connectionist learning models. One thing connectionist learning models are good at is extracting statistical regularities in their inputs. Indeed, it is sometimes tempting to say that connectionism in psychology is statistics for the non-statistician. As I mentioned earlier, it is often hard to work out how such models work, but it is every bit as important to discover why they work. For example, if they work because they compute statistics (e.g., the statistics relating spelling to pronunciation, or semantic features to words), then unless one wishes to argue that the particular connectionist algorithm like back-propagation (Rummelhart, Hinton and Williams, 1986) is a true reflection of the way computation of these statistics is implemented in the brain, the underlying explanation is really that behaviour is driven by some process that computes the relevant statistics. This is by no means a trivial claim. If some particular behaviour would emerge from almost any system that computed the right statistics, this tells us something very important. It tells us how much specialist pre-wired knowledge and architecture we need to build into the system.

CONCLUSIONS

A good model must be judged in terms of what it contributes to the explanation of the phenomena under investigation. Modelling really is a good thing, but not an end in itself. The first benefit of modelling is that it may focus your attention on details of the theory that you wouldn't otherwise have thought about. To produce a working computer program you may have to revise your theory, or even abandon it altogether. If I had to rank order the benefits of computational modelling, "Modelling makes you think" would be at the top of the list. But, even if you manage to produce a model that can simulate the data, that isn't the end of the story. Just having a model won't automatically explain how things work or, more importantly, why things work the way they do. As Roberts and Pashler note, a theory that could explain almost any pattern of data wouldn't be very illuminating. If you are successful at building a model, and that model was designed as a straightforward computational implementation of a carefully formulated theory, you need to make sure you understand exactly how the model relates to the theory. Does the model work solely because it instantiates the assumptions in the theory, or does it work because of something that was added to implement the model? Next comes the really important bit. Once you have established that the model does the right thing, for the right reasons, then you need

to ask what it is about the model that enables it to simulate the data. Why does this model work? Is there something special about exact computations your model performs? Perhaps there is nothing much about the specifics of your model that makes it work. Maybe the model just happens to incorporate the right set of principles, and any model that incorporated those principles would do the job. Discovering that yours isn't the one true model might not sound too satisfying, but it is likely to mean that you are on the way to having an explanation. You will have begun to explain how and why things work. Simulations from the model will convince you (and maybe even your critics) that the theory makes the right predictions, but it is only by thinking about the model that you will be able to explain why things work the way they do.

REFERENCES

Allopenna, P. D., Magnuson, J. S., & Tanenhaus, M. K. (1998). Tracking the time course of spoken word recognition using eye movements: Evidence for continuous mapping models. *Journal of Memory and Language, 38*, 419-439.

Brown, G. D., Preece, T., & Hulme, C. (2000). Oscillator-based memory for serial order. *Psychological Review, 107*, 127-181.

Burgess, N., & Hitch, G. J. (1992). Toward a Network Model of the Articulatory Loop. *Journal of Memory and Language, 31*, 429-460.

Burgess, N., & Hitch, G. J. (1999). Memory for serial order: A network model of the phonological loop and its timing. *Psychological Review, 106*, 551-581.

Forster, K. I. (1994). Computational modeling and elementary process analysis in visual word recognition. *Journal of Experimental Psychology: Human Perception and Performance, 20*, 1292-1310.

Grossberg, S. (1987). Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, 11, 23-63.

Henson, R. N. A. (1998). Short-term memory for serial order: The start-end model. *Cognitive Psychology, 36*, 73-137.

Lee, C. L., & Estes, W. K. (1977). Order and position in primary memory for letter strings. *Journal of Verbal Learning and Verbal Behaviour, 6,* 395-418.

Lee, C. L., & Estes, W. K. (1981). Item and order information in short-term memory: Evidence for multilevel perturbation processes. *Journal of Experimental Psychology: Human Learning and Memory, 7,* 149-169.

Marr, D. (1982). *Vision: A computational investigation into the human representation and processing of visual information*. San Francisco: Freeman & Co.

Marslen-Wilson, W. D. (1987). Functional parallelism in spoken word-recognition. *Cognition, 25,* 71-102.

Marslen-Wilson, W. D., & Welsh, A. (1978). Processing interactions and lexical access during word-recognition in continuous speech. *Cognitive Psychology, 10,* 29-63.

McClelland, J. L., & Elman, J. L. (1986). The TRACE model of speech perception. *Cognitive Psychology, 18*, 1-86.

McCloskey, M. (1991). Networks and theories: The place of connectionism in cognitive science. *Psychological Science, 2*, 387-395.

McQueen, J. M., Norris, D., & Cutler, A. (1994). Competition in spoken word recognition: Spotting words in other words. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 20*, 621-638.

Norris, D. (1994). Shortlist: A connectionist model of continuous speech recognition. *Cognition, 52*, 189-234.

Norris, D., McQueen, J. M., & Cutler, A. (1995). Competition and segmentation in spoken-word recognition. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 21*, 1209-1228.

Norris, D., McQueen, J. M., Cutler, A., & Butterfield, S. (1997). The possible-word constraint in the segmentation of continuous speech. *Cognitive Psychology, 34*, 191-243.

Norris, D., McQueen, J. M., & Cutler, A. (2000a). Merging information in speech recognition: Feedback is never necessary. *Behavioral and Brain Sciences, 23*, 299-370.

Norris, D., McQueen, J. M., & Cutler, A. (2000b). Feedback on feedback on feedback: It's feedforward - Authors' response. *Behavioral and Brain Sciences, 23*, 352-370.

Page, M. P. A., & Norris, D. (1998). The primacy model: A new model of immediate serial recall. *Psychological Review, 105*, 761-781.

Ratcliff, R. (1978). A theory of memory retrieval. *Psychological Review, 85*, 59-109.

Roberts, S., & Pashler, H. (2000). How persuasive is a good fit? A comment on theory testing. *Psychological Review*, *107*, 358-367.

Roberts, S., & Pashler, H. (2002). Theory development should begin (but not end) with good empirical fits: A comment on Roberts and Pashler (2000) - Reply to Rodgers and Rowe (2002). *Psychological Review, 109*, 605-607.

Ruchkin, D. S., Grafman, J., Cameron, K., & Berndt, R. S. (in press). Working memory retention systems: A state of activated long-term memory. *Brain and Behavioral Sciences.*

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D.E. Rumelhart & J.L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition,* Vol. 1 (pp. 318-362). Cambridge, MA: MIT Press.

Tanenhaus, M. K., Magnuson, J. S., McMurray, B., & Aslin, R. N. (2002). No compelling evidence against feedback in spoken word recognition. *Behavioral and Brain Sciences, 23*, 348-349.

Scharenborg, O., Norris, D., ten Bosch, L., & McQueen, J. M. (in prep). How should an optimal speech recognizer work?

Simon, H. A. (2002). Computational Models: Why build them? In L. Nadel (Ed.), *Encyclopaedia of Cognitive Science* (pp. 616-624). London**:** McMillan.

Vallar, G., & Baddeley, A. D. (1984). Fractionation of working memory: Neuropsychological evidence for a phonological short-term store. *Journal of Verbal Learning and Verbal Behavior*, 23, 151-161.