CHAPTER 18

On Learning the Past Tenses of English Verbs

D. E. RUMELHART and J. L. McCLELLAND

THE ISSUE

Scholars of language and psycholinguistics have been among the first to stress the importance of rules in describing human behavior. The reason for this is obvious. Many aspects of language can be characterized by rules, and the speakers of natural languages speak the language correctly. Therefore, systems of rules are useful in characterizing what they will and will not say. Though we all make mistakes when we speak, we have a pretty good ear for what is right and what is wrong—and our judgments of correctness—or grammaticality—are generally even easier to characterize by rules than actual utterances.

On the evidence that what we will and won't say and what we will and won't accept can be characterized by rules, it has been argued that, in some sense, we "know" the rules of our language. The sense in which we know them is not the same as the sense in which we know such "rules" as "*i* before e except after *c*," however, since we need not necessarily be able to state the rules explicitly. We know them in a way that allows us to use them to make judgments of grammaticality, it is often said, or to speak and understand, but this knowledge is not in a form or location that permits it to be encoded into a communicable verbal statement. Because of this, this knowledge is said to be *implicit*.

A slight variant of this chapter will appear in B. MacWhinney (Ed.), *Mechanisms of language acquisition*. Hillsdale, NJ: Erlbaum (in press).

So far there is considerable agreement. However, the exact characterization of implicit knowledge is a matter of great controversy. One view, which is perhaps extreme but is nevertheless quite clear, holds that the rules of language are stored in explicit form as propositions, and are used by language production, comprehension, and judgment mechanisms. These propositions cannot be described verbally only because they are sequestered in a specialized subsystem which is used in language processing, or because they are written in a special code that only the language processing system can understand. This view we will call the *explicit inaccessible rule* view.

On the explicit inaccessible rule view, language acquisition is thought of as the process of inducing rules. The language mechanisms are thought to include a subsystem—often called the *language acquisition device* (LAD)—whose business it is to discover the rules. A considerable amount of effort has been expended on the attempt to describe how the LAD might operate, and there are a number of different proposals which have been laid out. Generally, though, they share three assumptions:

- The mechanism hypothesizes explicit inaccessible rules.
- Hypotheses are rejected and replaced as they prove inadequate to account for the utterances the learner hears.
- The LAD is presumed to have *innate* knowledge of the possible range of human languages and, therefore, is presumed to consider only hypotheses within the constraints imposed by a set of *linguistic universals*.

The recent book by Pinker (1984) contains a state-of-the-art example of a model based on this approach.

We propose an alternative to explicit inaccessible rules. We suggest that lawful behavior and judgments may be produced by a mechanism in which there is no explicit representation of the rule. Instead, we suggest that the mechanisms that process language and make judgments of grammaticality are constructed in such a way that their performance is characterizable by rules, but that the rules themselves are not written in explicit form anywhere in the mechanism. An illustration of this view, which we owe to Bates (1979), is provided by the honeycomb. The regular structure of the honeycomb arises from the interaction of forces that wax balls exert on each other when compressed. The honeycomb can be described by a rule, but the mechanism which produces it does not contain any statement of this rule.

In our earlier work with the interactive activation model of word perception (McClelland & Rumelhart, 1981; Rumelhart & McClelland,

1981, 1982), we noted that lawful behavior emerged from the interactions of a set of word and letter units. Each word unit stood for a particular word and had connections to units for the letters of the word. There were no separate units for common letter clusters and no explicit provision for dealing differently with orthographically regular letter sequences—strings that accorded with the rules of English—as opposed to irregular sequences. Yet the model did behave differently with orthographically regular nonwords than it behaved with words. In fact, the model simulated rather closely a number of results in the word perception literature relating to the finding that subjects perceive letters in orthographically regular letter strings more accurately than they perceive letters in irregular, random letter strings. Thus, the behavior of the model was lawful even though it contained no explicit rules.

It should be said that the pattern of perceptual facilitation shown by the model did not correspond exactly to any system of orthographic rules that we know of. The model produced as much facilitation, for example, for special nonwords like *SLNT*, which are clearly irregular, as it did for matched regular nonwords like *SLET*. Thus, it is not correct to say that the model exactly mimicked the behavior we would expect to emerge from a system which makes use of explicit orthographic rules. However, neither do human subjects. Just like the model, they showed equal facilitation for vowelless strings like *SLNT* as for regular nonwords like *SLET*. Thus, human perceptual performance seems, in this case at least, to be characterized only approximately by rules.

Some people have been tempted to argue that the behavior of the model shows that we can do without linguistic rules. We prefer, however, to put the matter in a slightly different light. There is no denying that rules still provide a fairly close characterization of the performance of our subjects. And we have no doubt that rules are even more useful in characterizations of sentence production, comprehension, and grammaticality judgments. We would only suggest that parallel distributed processing models may provide a mechanism sufficient to capture lawful behavior, without requiring the postulation of explicit but inaccessible rules. Put succinctly, our claim is that PDP models provide an alternative to the explicit but inaccessible rules account of implicit knowledge of rules.

We can anticipate two kinds of arguments against this kind of claim. The first kind would claim that although certain types of rule-guided behavior might emerge from PDP models, the models simply lack the computational power needed to carry out certain types of operations which can be easily handled by a system using explicit rules. We believe that this argument is simply mistaken. We discuss the issue of computational power of PDP models in Chapter 4. Some applications of PDP models to sentence processing are described in Chapter 19.

The second kind of argument would be that the details of language behavior, and, indeed, the details of the language acquisition process, would provide unequivocal evidence in favor of a system of explicit rules.

It is this latter kind of argument we wish to address in the present chapter. We have selected a phenomenon that is often thought of as demonstrating the acquisition of a linguistic rule. And we have developed a parallel distributed processing model that learns in a natural way to behave in accordance with the rule, mimicking the general trends seen in the acquisition data.

THE PHENOMENON

The phenomenon we wish to account for is actually a sequence of three stages in the acquisition of the use of past tense by children learning English as their native tongue. Descriptions of development of the use of the past tense may be found in Brown (1973), Ervin (1964), and Kuczaj (1977).

In Stage 1, children use only a small number of verbs in the past tense. Such verbs tend to be very high-frequency words, and the majority of these are irregular. At this stage, children tend to get the past tenses of these words correct if they use the past tense at all. For example, a child's lexicon of past-tense words at this stage might consist of *came, got, gave, looked, needed, took*, and *went*. Of these seven verbs, only two are regular—the other five are generally idiosyncratic examples of irregular verbs. In this stage, there is no evidence of the use of the rule—it appears that children simply know a small number of separate items.

In Stage 2, evidence of implicit knowledge of a linguistic rule emerges. At this stage, children use a much larger number of verbs in the past tense. These verbs include a few more irregular items, but it turns out that the majority of the words at this stage are examples of the *regular* past tense in English. Some examples are *wiped* and *pulled*.

The evidence that the Stage 2 child actually has a linguistic rule comes not from the mere fact that he or she knows a number of regular forms. There are two additional and crucial facts:

- The child can now generate a past tense for an invented word. For example, Berko (1958) has shown that if children can be convinced to use *rick* to describe an action, they will tend to say *ricked* when the occasion arises to use the word in the past tense.

- Children now *incorrectly* supply regular past-tense endings for words which they used correctly in Stage 1. These errors may involve either adding *ed* to the root as in *comed* /kˆmd/, or adding *ed* to the irregular past tense form as in *camed* /kAmd/[1] (Ervin, 1964; Kuczaj, 1977).

Such findings have been taken as fairly strong support for the assertion that the child at this stage has acquired the past-tense "rule." To quote Berko (1958):

> If a child knows that the plural of *witch* is *witches*, he may simply have memorized the plural form. If, however, he tells us that the plural of *gutch* is *gutches*, we have evidence that he actually knows, albeit unconsciously, one of those rules which the descriptive linguist, too, would set forth in his grammar. (p. 151)

In Stage 3, the regular and irregular forms coexist. That is, children have regained the use of the correct irregular forms of the past tense, while they continue to apply the regular form to new words they learn. Regularizations persist into adulthood—in fact, there is a class of words for which either a regular or an irregular version are both considered acceptable—but for the commonest irregulars such as those the child acquired first, they tend to be rather rare. At this stage there are some clusters of exceptions to the basic, regular past-tense pattern of English. Each cluster includes a number of words that undergo identical changes from the present to the past tense. For example, there is a *ing/ang* cluster, an *ing/ung* cluster, an *eet/it* cluster, etc. There is also a group of words ending in /d/ or /t/ for which the present and past are identical.

Table 1 summarizes the major characteristics of the three stages.

Variability and Gradualness

The characterization of past-tense acquisition as a sequence of three stages is somewhat misleading. It may suggest that the stages are clearly demarcated and that performance in each stage is sharply distinguished from performance in other stages.

---

[1] The notation of phonemes used in this chapter is somewhat nonstandard. It is derived from the computer-readable dictionary containing phonetic transcriptions of the verbs used in the simulations. A key is given in Table 5.

TABLE 1

CHARACTERISTICS OF THE THREE STAGES OF PAST TENSE ACQUISITION

Verb Type	Stage 1	Stage 2	Stage 3
Early Verbs	Correct	Regularized	Correct
Regular	—	Correct	Correct
Other Irregular	—	Regularized	Correct or Regularized
Novel	—	Regularized	Regularized

In fact, the acquisition process is quite gradual. Little detailed data exists on the transition from Stage 1 to Stage 2, but the transition from Stage 2 to Stage 3 is quite protracted and extends over several years (Kuczaj, 1977). Further, performance in Stage 2 is extremely variable. Correct use of irregular forms is never completely absent, and the same child may be observed to use the correct past of an irregular, the base+ed form, and the past+ed form, within the same conversation.

Other Facts About Past-Tense Acquisition

Beyond these points, there is now considerable data on the detailed types of errors children make throughout the acquisition process, both from Kuczaj (1977) and more recently from Bybee and Slobin (1982). We will consider aspects of these findings in more detail below. For now, we mention one intriguing fact: According to Kuczaj (1977), there is an interesting difference in the errors children make to irregular verbs at different points in Stage 2. Early on, regularizations are typically of the base+ed form, like *goed*; later on, there is a large increase in the frequency of past+ed errors, such as *wented*.

THE MODEL

The goal of our simulation of the acquisition of past tense was to simulate the three-stage performance summarized in Table 1, and to see whether we could capture other aspects of acquisition. In particular, we wanted to show that the kind of gradual change characteristic of normal acquisition was also a characteristic of our distributed model, and we wanted to see whether the model would capture detailed aspects

of the phenomenon, such as the change in error type in later phases of development and the change in differences in error patterns observed for different types of words.

We were not prepared to produce a full-blown language processor that would learn the past tense from full sentences heard in everyday experience. Rather, we have explored a very simple past-tense learning environment designed to capture the essential characteristics necessary to produce the three stages of acquisition. In this environment, the model is presented, as learning experiences, with pairs of inputs—one capturing the phonological structure of the root form of a word and the other capturing the phonological structure of the correct past-tense version of that word. The behavior of the model can be tested by giving it just the root form of a word and examining what it generates as its "current guess" of the corresponding past-tense form.

Structure of the Model

The basic structure of the model is illustrated in Figure 1. The model consists of two basic parts: (a) a simple *pattern associator* network similar to those studied by Kohonen (1977; 1984; see Chapter 2) which learns the relationships between the base form and the past-tense

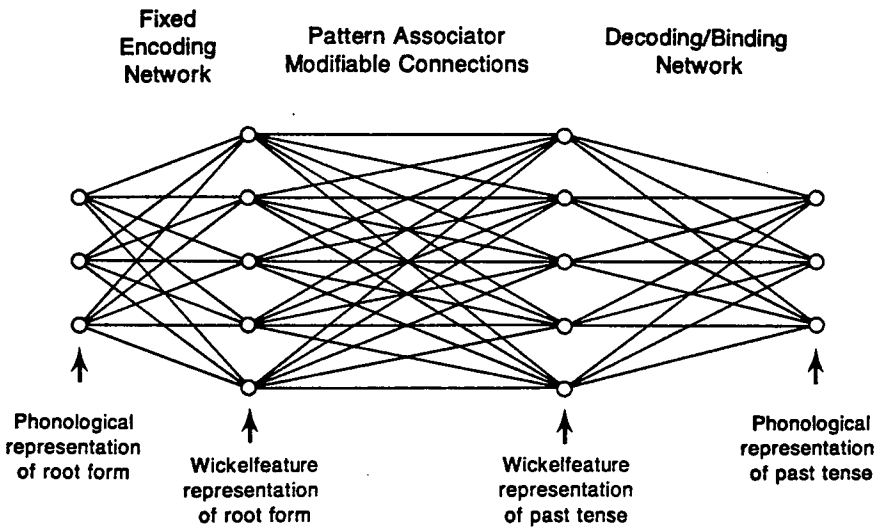


FIGURE 1. The basic structure of the model.

form, and (b) a decoding network that converts a featural representation of the past-tense form into a phonological representation. All learning occurs in the pattern associator; the decoding network is simply a mechanism for converting a featural representation which may be a near miss to any phonological pattern into a legitimate phonological representation. Our primary focus here is on the pattern associator. We discuss the details of the decoding network in the Appendix.

Units. The pattern associator contains two pools of units. One pool, called the input pool, is used to represent the input pattern corresponding to the root form of the verb to be learned. The other pool, called the output pool, is used to represent the output pattern generated by the model as its current guess as to the past tense corresponding to the root form represented in the inputs.

Each unit stands for a particular feature of the input or output string. The particular features we used are important to the behavior of the model, so they are described in a separate section below.

Connections. The pattern associator contains a modifiable connection linking each input unit to each output unit. Initially, these connections are all set to 0 so that there is no influence of the input units on the output units. Learning, as in other PDP models described in this book, involves modification of the strengths of these interconnections, as described below.

Operation of the Model

On test trials, the simulation is given a phoneme string corresponding to the root of a word. It then performs the following actions. First, it encodes the root string as a pattern of activation over the input units. The encoding scheme used is described below. Node activations are discrete in this model, so the activation values of all the units that should be on to represent this word are set to 1, and all the others are set to 0. Then, for each output unit, the model computes the net input to it from all of the weighted connections from the input units. The net input is simply the sum over all input units of the input unit activation times the corresponding weight. Thus, algebraically, the net input to output unit i is

$$net_i = \sum_j a_j w_{ij}$$

where a_j represents the activation of input unit j, and w_{ij} represents the weight from unit j to unit i.

Each unit has a threshold, θ, which is adjusted by the learning procedure that we will describe in a moment. The probability that the unit is turned on depends on the amount the net input exceeds the threshold. The *logistic* probability function is used here as in the Boltzmann machine (Chapter 7) and in harmony theory (Chapter 6) to determine whether the unit should be turned on. The probability is given by

$$p\ (a_i = 1) = \frac{1}{1 + e^{-(net_i - \theta_i)/T}} \tag{1}$$

where T represents the temperature of the system. The logistic function is shown in Figure 2. The use of this probabilistic response rule allows the system to produce different responses on different occasions with the same network. It also causes the system to learn more slowly so the effect of regular verbs on the irregulars continues over a much longer period of time. As discussed in Chapter 2, the temperature, T, can be manipulated so that at very high temperatures the response of the units is highly variable; with lower values of T, the units behave more like *linear threshold units.*

Since the pattern associator built into the model is a one-layer net with no feedback connections and no connections from one input unit to another or from one output unit to another, iterative computation is of no benefit. Therefore, the processing of an input pattern is a simple matter of first calculating the net input to each output unit and then

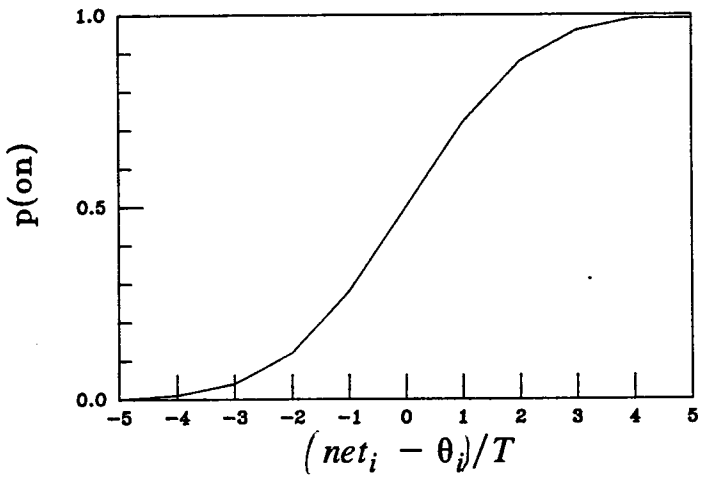


FIGURE 2. The logistic function used to calculate probability of activation. The x-axis shows values of $net_i - \theta_i/T$, and the y-axis indicates the corresponding probability that unit i will be activated.

setting its activation probabilistically on the basis of the logistic equation given above. The temperature T only enters in setting the variability of the output units; a fixed value of T was used throughout the simulations.

To determine how well the model did at producing the correct output, we simply compare the pattern of output Wickelphone activations to the pattern that the correct response would have generated. To do this, we first translate the correct response into a target pattern of activation for the output units, based on the same encoding scheme used for the input units. We then compare the obtained pattern with the target pattern on a unit-by-unit basis. If the output perfectly reproduces the target, then there should be a 1 in the output pattern wherever there is a 1 in the target. Such cases are called *hits*, following the conventions of signal detection theory (Green & Swets, 1966). There should also be a 0 in the output whenever there is a 0 in the target. Such cases are called *correct rejections*. Cases in which there are 1s in the output but not in the target are called *false alarms*, and cases in which there are 0s in the output that should be present in the input are called *misses*. A variety of measures of performance can be computed. We can measure the percentage of output units that match the correct past tense, or we can compare the output to the pattern for any other response alternative we might care to evaluate. This allows us to look at the output of the system independently of the decoding network. We can also employ the decoding network and have the system synthesize a phonological string. We can measure the performance of the system either at the featural level or at the level of strings of phonemes. We shall employ both of these mechanisms in the evaluation of different aspects of the overall model.

Learning

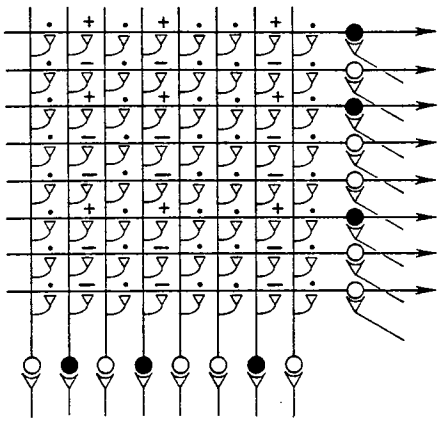
On a learning trial, the model is presented with both the root form of the verb and the target. As on a test trial, the pattern associator network computes the output it would generate from the input. Then, for each output unit, the model compares its answer with the target. Connection strengths are adjusted using the classic *perceptron convergence procedure* (Rosenblatt, 1962). The perceptron convergence procedure is simply a discrete variant of the delta rule presented in Chapter 2 and discussed in many places in this book. The exact procedure is as follows: We can think of the target as supplying a teaching input to each output unit, telling it what value it ought to have. When the actual output matches the target output, the model is doing the right thing

and so none of the weights on the lines coming into the unit are adjusted. When the computed output is 0 and the target says it should be 1, we want to increase the probability that the unit will be active the next time the same input pattern is presented. To do this, we increase the weights from all of the input units that are active by a small amount η. At the same time, the threshold is also reduced by η. When the computed output is 1 and the target says it should be 0, we want to decrease the probability that the unit will be active the next time the same input pattern is presented. To do this, the weights from all of the input units that are active are reduced by η, and the threshold is increased by η. In all of our simulations, the value of η is simply set to 1. Thus, each change in a weight is a unit change, either up or down. For nonstochastic units, it is well known that the perceptron convergence procedure will find a set of weights that will allow the model to get each output unit correct, provided that such a set of weights exists. For the stochastic case, it is possible for the learning procedure to find a set of weights that will make the probability of error as low as desired. Such a set of weights exists if a set of weights exists that will always get the right answer for nonstochastic units.

Learning Regular and Exceptional Patterns in a Pattern Associator

In this section, we present an illustration of the behavior of a simple pattern associator model. The model is a scaled-down version of the main simulation described in the next section. We describe the scaled-down version first because in this model it is possible to actually examine the matrix of connection weights, and from this to see clearly how the model works and why it produces the basic three-stage learning phenomenon characteristic of acquisition of the past tense. Various aspects of pattern associator networks are described in a number of places in this book (Chapters 1, 2, 8, 9, 11, and 12, in particular) and elsewhere (J. A. Anderson, 1973, 1977; J. A. Anderson, Silverstein, Ritz, & Jones, 1977; Kohonen, 1977, 1984). Here we focus our attention on their application to the representation of rules for mapping one set of patterns into another.

For the illustration model, we use a simple network of eight input and eight output units and a set of connections from each input unit to each output unit. The network is illustrated in Figure 3. The network is shown with a set of connections sufficient for associating the pattern of activation illustrated on the input units with the pattern of activation illustrated on the output units. (Active units are darkened; positive



+15	+15					+15	
−15	−15					−15	
+15	+15					+15	
−15	−15					−15	
−15	−15					−15	
+15	+15					+15	
−15	−15					−15	
−15	−15					−15	

FIGURE 3. Simple network used in illustrating basic properties of pattern associator networks; excitatory and inhibitory connections needed to allow the active input pattern to produce the illustrated output pattern are indicated with + and −. Next to the network, the matrix of weights indicating the strengths of the connections from each input unit to each output unit. Input units are indexed by the column they appear in; output units are indexed by row.

and negative connections are indicated by numbers written on each connection). Next to the network is the matrix of connections abstracted from the actual network itself, with numerical values assigned to the positive and negative connections. Note that each weight is located in the matrix at the point where it occurred in the actual network diagram. Thus, the entry in the ith row of the jth column indicates the connection w_{ij} from the jth input unit to the ith output unit.

Using this diagram, it is easy to compute the net inputs that will arise on the output units when an input pattern is presented. For each output unit, one simply scans across its rows and adds up all the weights found in columns associated with active input units. (This is exactly what the simulation program does!) The reader can verify that when the input pattern illustrated in the left-hand panel is presented, each output unit that should be on in the output pattern receives a net input of +45; each output unit that should be off receives a net input of −45.[2] Plugging these values into Equation 1, using a temperature

[2] In the examples we will be considering in this section, the thresholds of the units are fixed at 0. Threshold terms add an extra degree of freedom for each output unit and allow the unit to come on in the absence of input, but they are otherwise inessential to the operation of the model. Computationally, they are equivalent to an adjustable weight to an extra input unit that is always on.

of 15,[3] we can compute that each output unit will take on the correct value about 95% of the time. The reader can check this in Figure 2; when the net input is +45, the exponent in the denominator of the logistic function is 3, and when the net input is −45, the exponent is −3. These correspond to activation probabilities of about .95 and .05, respectively.

One of the basic properties of the pattern associator is that it can store the connections appropriate for mapping a number of different input patterns to a number of different output patterns. The perceptron convergence procedure can accommodate a number of arbitrary associations between input patterns and output patterns, as long as the input patterns form a linearly independent set (see Chapters 9 and 11). Table 2 illustrates this aspect of the model. The first two cells of the table show the connections that the model learns when it is trained on each of the two indicated associations separately. The third cell shows connections learned by the model when it is trained on both patterns in alternation, first seeing one and then seeing the other of the two. Again, the reader can verify that if either input pattern is presented to a network with this set of connections, the correct corresponding output pattern is reconstructed with high probability; each output unit that should be on gets a net input of at least +45, and each output unit that should be off gets a net input below −45.

The restriction of networks such as this to linearly independent sets of patterns is a severe one since there are only N linearly independent patterns of length N. That means that we could store at most eight unrelated associations in the network and maintain accurate performance. However, if the patterns all conform to a general rule, the capacity of the network can be greatly enhanced. For example, the set of connections shown in Table 2D is capable of processing all of the patterns defined by what we call the *rule of 78*. The rule is described in Table 3. There are 18 different input/output pattern pairs corresponding to this rule, but they present no difficulty to the network. Through repeated presentations of examples of the rule, the perceptron convergence procedure learned the set of weights shown in cell D of Table 2. Again, the reader can verify that it works for any legal association fitting the rule of 78. (Note that for this example, the "regular" pairing

[3] For the actual simulations of verb learning, we used a value of T equal to 200. This means that for a fixed value of the weight on an input line, the effect of that line being active on the unit's probability of firing is much lower than it is in these illustrations. This is balanced by the fact that in the verb learning simulations, a much larger number of inputs contribute to the activation of each output unit. Responsibility for turning a unit on is simply more distributed when larger input patterns are used.

TABLE 2

WEIGHTS IN THE 8-UNIT NETWORK
AFTER VARIOUS LEARNING EXPERIENCES

A. Weights acquired in learning (2 4 7) → (1 4 6)

.	15	.	15	.	.	15	.
.	-16	.	-16	.	.	-16	.
.	-17	.	-17	.	.	-17	.
.	16	.	16	.	.	16	.
.	-16	.	-16	.	.	-16	.
.	17	.	17	.	.	17	.
.	-16	.	-16	.	.	-16	.
.	-17	.	-17	.	.	-17	.

B. Weights acquired in learning (3 4 6) → (3 6 7)

.	.	.	-16	-16	.	-16	.
.	.	.	-17	-17	.	-17	.
.	.	.	17	17	.	17	.
.	.	.	-16	-16	.	-16	.
.	.	.	-17	-17	.	-17	.
.	.	.	16	16	.	16	.
.	.	.	17	17	.	17	.
.	.	.	-17	-17	.	-17	.

C. Weights acquired in learning A and B together

.	24	-24	.	.	-24	24	.
.	-13	-13	-26	.	-13	-13	.
.	-23	24	1	.	24	-23	.
.	24	-25	-1	.	-25	24	.
.	-13	-13	-26	.	-13	-13	.
.	13	13	26	.	13	13	.
.	-25	24	-1	.	24	-25	.
.	-12	-13	-25	.	-13	-12	.

D. Weights acquired in learning the rule of 78

61	-37	-37	-5	-5	-3	-6	-7
-35	60	-38	-4	-6	-3	-5	-8
-39	-35	61	-4	-5	-4	-7	-6
-6	-4	-5	59	-37	-37	-8	-7
-5	-5	-4	-36	60	-38	-7	-7
-5	-4	-6	-37	-38	60	-8	-7
.	1	.	1	.	.	-50	51
.	.	-1	-2	1	.	49	-50

TABLE 3

THE RULE OF 78

Input patterns consist of one active unit from each of the following sets:	(1 2 3) (4 5 6) (7 8)
The output pattern paired with a given input pattern consists of:	The same unit from (1 2 3) The same unit from (4 5 6) The other unit from (7 8)
Examples:	2 4 7 → 2 4 8 1 6 8 → 1 6 7 3 5 7 → 3 5 8
An exception:	1 4 7 → 1 4 7

of (1 4 7) with (1 4 8) was used rather than the exceptional mapping illustrated in Table 3).

We have, then, observed an important property of the pattern associator: If there is some structure to a set of patterns, the network may be able to learn to respond appropriately to all of the members of the set. This is true, even though the input vectors most certainly do not form a linearly independent set. The model works anyway because the response that the model should make to some of the patterns can be predicted from the responses that it should make to others of the patterns.

Now let's consider a case more like the situation a young child faces in learning the past tenses of English verbs. Here, there is a regular pattern, similar to the rule of 78. In addition, however, there are exceptions. Among the first words the child learns are many exceptions, but as the child learns more and more verbs, the proportion that are regular increases steadily. For an adult, the vast majority of verbs are regular.

To examine what would happen in a pattern associator in this kind of a situation, we first presented the illustrative 8-unit model with two pattern pairs. One of these was a regular example of the 78 rule [(2 5 8)→(2 5 7)]. The other was an exception to the rule [(1 4 7)→(1 4 7)]. The simulation saw both pairs 20 times, and connection strengths were adjusted after each presentation. The resulting set of connections is shown in cell A of Table 4. This number of learning trials is not enough to lead to perfect performance; but after this much experience, the model tends to get the right answer for each output unit close to 90 percent of the time. At this point, the fact that one of the patterns is an example of a general rule and the other is an exception to that rule is irrelevant to the model. It learns a set of connections that can accommodate these two patterns, but it cannot generalize to new instances of the rule.

This situation, we suggest, characterizes the situation that the language learner faces early on in learning the past tense. The child knows, at this point, only a few high-frequency verbs, and these tend, by and large, to be irregular, as we shall see below. Thus each is treated by the network as a separate association, and very little generalization is possible.

But as the child learns more and more verbs, the proportion of regular verbs increases. This changes the situation for the learning model. Now the model is faced with a number of examples, all of which follow the rule, as well as a smattering of irregular forms. This new situation changes the experience of the network, and thus the pattern of interconnections it contains. Because of the predominance of the regular

TABLE 4

REPRESENTING EXCEPTIONS: WEIGHTS IN THE 8-UNIT NETWORK

A. After 20 exposures to (1 4 7)→(1 4 7), (2 5 8)→(2 5 7)

12	-12	.	12	-12	.	12	-12
-11	13	.	-11	13	.	-11	13
-11	-11	.	-11	-11	.	-11	-11
12	-12	.	12	-12	.	12	-12
-11	11	.	-11	11	.	-11	11
-11	-12	.	-11	-12	.	-11	-12
12	11	.	12	11	.	12	11
-11	-13	.	-11	-13	.	-11	-13

B. After 10 more exposures to all 18 associations

44	-34	-26	-2	-10	-4	-8	-8
-32	46	-27	-11	2	-4	-9	-4
-30	-24	43	-5	-5	-1	-2	-9
-1	-7	-7	45	-34	-26	-4	-11
-8	-3	-3	-31	44	-27	-7	-7
-6	-8	-3	-31	-28	42	-7	-10
11	-2	-6	11	-2	-6	-35	38
-9	-4	7	-13	1	6	36	-42

C. After 30 more exposures to all 18 associations

61	-38	-38	-6	-5	-4	-6	-9
-38	62	-39	-6	-5	-4	-8	-7
-37	-38	62	-5	-5	-3	-7	-6
-4	-6	-6	62	-40	-38	-8	-8
-5	-5	-4	-38	62	-38	-7	-7
-6	-4	-5	-38	-39	62	-8	-7
20	-5	-4	22	-5	-6	-50	61
-19	8	5	-18	5	7	54	-60

D. After a total of 500 exposures to all 18 associations

64	-39	-39	-5	-4	-5	-7	-7
-39	63	-39	-5	-5	-5	-7	-8
-39	-40	64	-5	-5	-5	-8	-7
-5	-5	-5	64	-40	-39	-8	-7
-5	-5	-5	-39	63	-39	-7	-8
-5	-5	-5	-39	-39	63	-8	-7
71	-28	-29	70	-28	-28	-92	106
-70	27	28	-70	27	28	91	-106

form in the input, the network learns the regular pattern, temporarily "overregularizing" exceptions that it may have previously learned.

Our illustration takes this situation to an extreme, perhaps, to illustrate the point. For the second stage of learning, we present the model with the entire set of eighteen input patterns consisting of one active unit from (1 2 3), one from (4 5 6), and one from (7 8). All of these patterns are regular except the one exception already used in the first stage of training.

At the end of 10 exposures to the full set of 18 patterns, the model has learned a set of connection strengths that predominantly captures the "regular pattern." At this point, its response to the exceptional pattern is *worse* than it was before the beginning of Phase 2; rather than getting the right output for Units 7 and 8, the network is now *regularizing* it.

The reason for this behavior is very simple. All that is happening is that the model is continually being bombarded with learning experiences directing it to learn the rule of 78. On only one learning trial out of 18 is it exposed to an exception to this rule.

In this example, the deck has been stacked very strongly against the exception. For several learning cycles, it is in fact quite difficult to tell from the connections that the model is being exposed to an exception mixed in with the regular pattern. At the end of 10 cycles, we can see that the model is building up extra excitatory connections from input Units 1 and 4 to output Unit 7 and extra inhibitory strength from Units 1 and 4 to Unit 8, but these are not strong enough to make the model get the right answer for output Units 7 and 8 when the (1 4 7) input pattern is shown. Even after 40 trials (panel C of Table 4), the model still gets the wrong answer on Units 7 and 8 for the (1 4 7) pattern more than half the time. (The reader can still be checking these assertions by computing the net input to each output unit that would result from presenting the (1 4 7) pattern.)

It is only after the model has reached the stage where it is making very few mistakes on the 17 regular patterns that it begins to accommodate to the exception. This amounts to making the connection from Units 1 and 4 to output Unit 7 strongly excitatory and making the connections from these units to output Unit 8 strongly inhibitory. The model must also make several adjustments to other connections so that the adjustments just mentioned do not cause errors on regular patterns similar to the exceptions, such as (1 5 7), (2 4 7), etc. Finally, in panel D, after a total of 500 cycles through the full set of 18 patterns, the weights are sufficient to get the right answer nearly all of the time. Further improvement would be very gradual since the network makes errors so infrequently at this stage that there is very little opportunity for change.

It is interesting to consider for a moment how an association is represented in a model like this. We might be tempted to think of the representation of an association as the difference between the set of connection strengths needed to represent a set of associations that includes the association and the set of strengths needed to represent the same set excluding the association of interest. Using this definition, we see that the representation of a particular association is far from invariant. What this means is that learning that occurs in one situation (e.g., in which there is a small set of unrelated associations) does not necessarily transfer to a new situation (e.g., in which there are a number of regular associations). This is essentially why the early learning our illustrative model exhibits of the (1 4 7) → (1 4 7) association in the context of just one other association can no longer support correct performance when the larger ensemble of regular patterns is introduced.

Obviously, the example we have considered in this section is highly simplified. However, it illustrates several basic facts about pattern associators. One is that they tend to exploit regularity that exists in the mapping from one set of patterns to another. Indeed, this is one of the

main advantages of the use of distributed representations. Second, they allow exceptions and regular patterns to coexist in the same network. Third, if there is a predominant regularity in a set of patterns, this can swamp exceptional patterns until the set of connections has been acquired that captures the predominant regularity. Then further, gradual tuning can occur that adjusts these connections to accommodate both the regular patterns and the exception. These basic properties of the pattern associator model lie at the heart of the three-stage acquisition process, and account for the gradualness of the transition from Stage 2 to Stage 3.

Featural Representations of Phonological Patterns

The preceding section describes basic aspects of the behavior of the pattern associator model and captures fairly well what happens when a pattern associator is applied to the processing of English verbs, following a training schedule similar to the one we have just considered for the acquisition of the rule of 78. There is one caveat, however: The input and target patterns—the base forms of the verbs and the correct past tenses of these verbs—must be represented in the model in such a way that the features provide a convenient basis for capturing the regularities embodied in the past-tense forms of English verbs. Basically, there were two considerations:

- We needed a representation that permitted a differentiation of all of the root forms of English and their past tenses.
- We wanted a representation that would provide a natural basis for generalizations to emerge about what aspects of a present tense correspond to what aspects of the past tense.

A scheme which meets the first criterion, but not the second, is the scheme proposed by Wickelgren (1969). He suggested that words should be represented as sequences of context-sensitive phoneme units, which represent each phone in a word as a triple, consisting of the phone itself, its predecessor, and its successor. We call these triples *Wickelphones*. Notationally, we write each Wickelphone as a triple of phonemes, consisting of the central phoneme, subscripted on the left by its predecessor and on the right by its successor. A phoneme occurring at the beginning of a word is preceded by a special symbol (#) standing for the word boundary; likewise, a phoneme occurring at the

end of a word is followed by #. The word /kat/, for example, would be represented as ${}_{\#}\mathbf{k}_{\mathbf{a}}$, ${}_{\mathbf{k}}\mathbf{a}_{\mathbf{t}}$, and ${}_{\mathbf{a}}\mathbf{t}_{\#}$. Though the Wickelphones in a word are not strictly position specific, it turns out that (a) few words contain more than one occurrence of any given Wickelphone, and (b) there are no two words we know of that consist of the same sequence of Wickelphones. For example, /slit/ and /silt/ contain no Wickelphones in common.

One nice property of Wickelphones is that they capture enough of the context in which a phoneme occurs to provide a sufficient basis for differentiating between the different cases of the past-tense rule and for characterizing the contextual variables that determine the subregularities among the irregular past-tense verbs. For example, the word-final phoneme that determines whether we should add /d/, /t/ or /ˆd/ in forming the regular past. And it is the sequence ${}_{\mathbf{i}}\mathbf{N}_{\#}$ which is transformed to ${}_{\mathbf{a}}\mathbf{N}_{\#}$ in the *ing* → *ang* pattern found in words like *sing*.

The trouble with the Wickelphone solution is that there are too many of them, and they are too specific. Assuming that we distinguish 35 different phonemes, the number of Wickelphones would be 35^3, or 42,875, not even counting the Wickelphones containing word boundaries. And, if we postulate one input unit and one output unit in our model for each Wickelphone, we require rather a large connection matrix (4.3×10^4 squared, or about 2×10^9) to represent all their possible connections.

Obviously, a more compact representation is required. This can be obtained by representing each Wickelphone as a distributed pattern of activation over a set of feature detectors. The basic idea is that we represent each phoneme, not by a single Wickelphone, but by a pattern of what we call *Wickelfeatures*. Each Wickelfeature is a conjunctive, or context-sensitive, feature, capturing a feature of the central phoneme, a feature of the predecessor, and a feature of the successor.

Details of the Wickelfeature representation. For concreteness, we will now describe the details of the feature coding scheme we used. It contains several arbitrary properties, but it also captures the basic principles of coarse, conjunctive coding described in Chapter 3. First, we will describe the simple feature representation scheme we used for coding a single phoneme as a pattern of features without regard to its predecessor and successor. Then we describe how this scheme can be extended to code whole Wickelphones. Finally, we show how we "blur" this representation, to promote generalization further.

To characterize each phoneme, we devised the highly simplified feature set illustrated in Table 5. The purpose of the scheme was (a) to give as many of the phonemes as possible a distinctive code, (b) to allow code similarity to reflect the similarity structure of the phonemes

TABLE 5

CATEGORIZATION OF PHONEMES ON FOUR SIMPLE DIMENSIONS

		Place					
		Front		Middle		Back	
		V/L	U/S	V/L	U/S	V/L	U/S
Interrupted	*Stop*	b	p	d	t	g	k
	Nasal	m	-	n	-	N	-
Cont. Consonant	*Fric.*	v/D	f/T	z	s	Z/j	S/C
	Liq/SV	w/l	-	r	-	y	h
Vowel	*High*	E	i	O	^	U	u
	Low	A	e	I	a/α	W	*/o

Key: N = ng in *sing*; D = th in *the*; T = th in *with*; Z = z in *azure*; S = sh in *ship*; C = ch in *chip*; E = ee in *beet*; i = i in *bit*; O = oa in *boat*; ^ = u in *but* or schwa; U = oo in *boot*; u = oo in *book*; A = ai in *bait*; e = e in *bet*; I = i_e in *bite*; a = a in *bat*; α = a in *father*; W = ow in *cow*; * = aw in *saw*; o = o in *hot*.

in a way that seemed sufficient for our present purposes, and (c) to keep the number of different features as small as possible.

The coding scheme can be thought of as categorizing each phoneme on each of four dimensions. The first dimension divides the phonemes into three major types: interrupted consonants (stops and nasals), continuous consonants (fricatives, liquids, and semivowels), and vowels. The second dimension further subdivides these major classes. The interrupted consonants are divided into plain stops and nasals; the continuous consonants into fricatives and sonorants (liquids and semivowels are lumped together); and the vowels into high and low. The third dimension classifies the phonemes into three rough places of articulation—front, middle, and back. The fourth subcategorizes the consonants into voiced vs. voiceless categories and subcategorizes the vowels into long and short. As it stands, the coding scheme gives identical codes to six pairs of phonemes, as indicated by the duplicate entries in the cells of the table. A more adequate scheme could easily be constructed by increasing the number of dimensions and/or values on the dimensions.

Using the above code, each phoneme can be characterized by one value on each dimension. If we assigned a unit for each value on each dimension, we would need 10 units to represent the features of a single phoneme since two dimensions have three values and two have two

values. We could then indicate the pattern of these features that corresponds to a particular phoneme as a pattern of activation over the 10 units.

Now, one way to represent each Wickelphone would simply be to use three sets of feature patterns: one for the phoneme itself, one for its predecessor, and one for its successor. To capture the word-boundary marker, we would need to introduce a special eleventh feature. Thus, the Wickelphone $_{\#}\mathbf{k}_{\mathbf{a}}$ can be represented by

[(000) (00) (000) (00) 1]
[(100) (10) (001) (01) 0]
[(001) (01) (010) (01) 0].

Using this scheme, a Wickelphone could be represented as a pattern of activation over a set of 33 units.

However, there is one drawback with this. The representation is not sufficient to capture more than one Wickelphone at a time. If we add another Wickelphone, the representation gives us no way of knowing which features belong together.

We need a representation, then, that provides us with a way of determining which features go together. This is just the job that can be done with detectors for Wickelfeatures—triples of features, one from the central phoneme, one from the predecessor phoneme, and one from the successor phoneme.

Using this scheme, each detector would be activated when the word contained a Wickelphone containing its particular combination of three features. Since each phoneme of a Wickelphone can be characterized by 11 features (including the word-boundary feature) and each Wickelphone contains three phonemes, there are $11 \times 11 \times 11$ possible Wickelfeature detectors. Actually, we are not interested in representing phonemes that cross word boundaries, so we only need 10 features for the center phoneme.

Though this leaves us with a fairly reasonable number of units ($11 \times 10 \times 11$ or 1,210), it is still large by the standards of what will easily fit in available computers. However, it is possible to cut the number down still further without much loss of representational capacity since a representation using all 1,210 units would be highly redundant; it would represent each feature of each of the three phonemes 16 different times, one for each of the conjunctions of that feature with one of the four features of one of the other phonemes and one of the four features of the other.

To cut down on this redundancy and on the number of units required, we simply eliminated all those Wickelfeatures specifying values on two different dimensions of the predecessor and the

successor phonemes. We kept all the Wickelfeature detectors for all combinations of different values on the same dimension for the predecessor and successor phonemes. It turns out that there are 260 of these (ignoring the word-boundary feature), and each feature of each member of each phoneme triple is still represented four different times. In addition, we kept the 100 possible Wickelfeatures combining a preceding word-boundary feature with any feature of the main phoneme and any feature of the successor; and the 100 Wickelfeatures combining a following word boundary feature with any feature of the main phoneme and any feature of the successor. All in all then, we used only 460 of the 1,210 possible Wickelfeatures.

Using this representation, a verb is represented by a pattern of activation over a set of 460 Wickelfeature units. Each Wickelphone activates 16 Wickelfeature units. Table 6 shows the 16 Wickelfeature units activated by the Wickelphone ${}_{k}A_{m}$, the central Wickelphone in the word *came*. The first Wickelfeature is turned on whenever we have a Wickelphone in which the preceding contextual phoneme is an interrupted consonant, the central phoneme is a vowel, and the following phoneme is an interrupted consonant. This Wickelfeature is turned on for the Wickelphone ${}_{k}A_{m}$ since /k/ and /m/, the context phonemes, are both interrupted consonants and /A/, the central phoneme, is a vowel. This same Wickelfeature would be turned on in the

TABLE 6

THE SIXTEEN WICKELFEATURES FOR THE WICKELPHONE ${}_{k}A_{m}$

Feature	Preceding Context	Central Phoneme	Following Context
1	Interrupted	Vowel	Interrupted
2	Back	Vowel	Front
3	Stop	Vowel	Nasal
4	Unvoiced	Vowel	Voiced
5	Interrupted	Front	Vowel
6	Back	Front	Front
7	Stop	Front	Nasal
8	Unvoiced	Front	Voiced
9	Interrupted	Low	Interrupted
10	Back	Low	Front
11	Stop	Low	Nasal
12	Unvoiced	Low	Voiced
13	Interrupted	Long	Vowel
14	Back	Long	Front
15	Stop	Long	Nasal
16	Unvoiced	Long	Voiced