(1986)
Terrence J. Sejnowski and Charles R. Rosenberg

NETtalk: a parallel network that learns to read aloud

The Johns Hopkins University Electrical Engineering and Computer Science Technical Report
JHU/EECS-86/01, 32 pp.

Unrestricted English text can be converted to speech by applying phonological rules and handling exceptions with a look-up table. However, this approach is highly labor intensive since each entry and rule must be hand-crafted. NETtalk is an alternative approach that is based on an automated learning procedure for a parallel network of deterministic processing units. After training on a corpus of informal continuous speech, it achieves good performance and generalizes to novel words. The distributed internal representations of the phonological regularities discovered by the network are damage resistant.

Introduction

English is amongst the most difficult languages to read aloud. Most phonological rules for transforming letters to speech sounds have exceptions that are often context-sensitive (1). For example, the "a" in almost all words ending in "ave", such as "brave" and "gave", is a long vowel, but-not in "have", and there are some words such as "survey" that can vary in pronunication with their syntactic role. The problem of reconciling rules and exceptions in converting text to speech shares some characteristics with difficult problems in artificial intelligence that have traditionally been approached with rule-based knowledge representations (2), such as natural language translation and abduction in expert systems (3).

DECtalk (4) is a commercial product that can produce intelligible speech synthesis in a restricted domain. DECtalk uses two methods for first converting text to phonemes (5): A word is first looked up in a pronunciation dictionary of common words; if it is not found there, then a set of phonological rules is applied. Phonemes and stress assignments are then converted into speech sounds using transition rules and digital speech synthesis. For novel words that are not correctly pronounced, this approach requires explicit intervention to update the dictionary and rules (6).

An alternative approach to knowledge representation is based on massively-parallel network models (7, 8). Knowledge in these models is distributed over many processing units and the behavior of the network in response to a particular input pattern is a collective decision based on the exchange of information between the processing units. These are called connectionist models because they are "programmed" by specifying the connectivity of the network and the strength or weight of each connection. In some cases the connections can be determined using insights from the problem domain, particularly when the networks have a regular pattern (9). It would be desireable to generate networks automatically, and one method is to "compile" a network from a description of a problem, such as a parser for a grammar (10). Another automatic method is incremental learning, which allows networks to be created by repetitive training on examples.

Rumelhart and McClelland (11) have successfully taught a one-layer network model to produce the past tenses of English verbs using the perceptron learning algorithm (12). The verb-learning network is rule-following but not rule-based in the sense that no rules are explicitly programmed into the network, but after training, the network behaves as if it were following rules. This is a consequence of the learning algorithm, which takes advantage of regularities of past tense endings to minimize the number of weights needed in the network, as shown by the ability of the network to generalize to novel verbs and pseudoverbs. However, a network with one layer of modifiable weights is severely restricted in its ability to discover higher-order features (13, 14).

Until recently, learning in multilayered networks was an unsolved problem and considered by some impossible (13, p. 231). In a multilayered machine the internal, or hidden, units can be used as feature detectors which perform a mapping between input units and output units, and the difficult problem is to discover the proper features. The Boltzmann machine learning algorithm is capable of finding features that allow a network to generalize from examples (8, 14, 15), and several other learning algorithms are now known for multilayered networks which can also discover good features (16, 17).

In this paper we explore the applicability of network learning algorithms with one to three layers of modifiable connections to the problem of converting text to

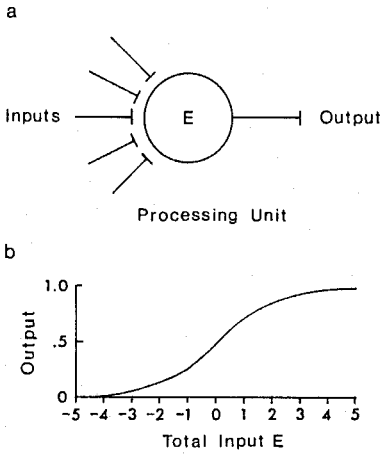


Figure 1 (a) Schematic model of a processing unit receiving inputs from other processing units. (b) Transformation between summed inputs and output of a processing unit, as given by Eq. 2.

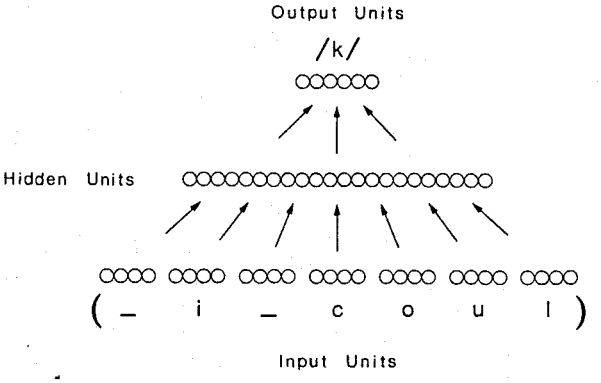


Figure 2 Schematic drawing of the network architecture. Input units are shown on the bottom of the pyramid, with 7 groups of 29 units in each group. Each hidden unit in the intermediate layer receives inputs from all of the input units on the bottom layer, and in turn sends its output to all 26 units in the output layer. An example of an input string of letters is shown below the inputs groups, and the correct output phoneme for the middle letter is shown above the output layer. For 80 hidden units, which were used for the corpus of continuous informal speech, there was a total of 309 units and 18,629 weights in the network, including a variable threshold for each unit.

speech. The model, which we call NETtalk, demonstrates that a relatively small network can capture most of the significant regularities in English pronunciation as well as absorb many of the irregularities. NETtalk can be trained on any dialect of any language and the resulting network can be implemented directly in hardware.

We will first describe the network architecture and the learning algorithm that we used, and then present the results obtained from simulations. Finally, we discuss the computational complexity of NETtalk and some of its biological implications.

Network Architecture

Processing Units

The network is composed of processing units that nonlinearly transform their summed, continuous-valued inputs, as illustrated in Fig. 1. The connection strength, or weight, linking one unit to another unit can be a positive or negative real value, representing either an excitatory or an inhibitory influence of the first unit on the output of the second unit. Each unit also has a threshold which is subtracted from the summed input. The threshold is implemented as weight from a unit that has a fixed value of 1 so that the same notation and learning algorithm can also be applied to the thresholds as well as the weights. The output of the *ith* unit is determined by first summing all of its inputs

$$E_i = \sum_j w_{ij} p_j \tag{1}$$

where w_{ij} is the weight from the *jth* to the *ith* unit, and then applying a sigmoidal transformation

$$p_i = P(E_i) = \frac{1}{1 + e^{-E_i}} \tag{2}$$

as shown in Fig. 1.

The network used in NETtalk is hierarchically arranged into three layers of units: an input layer, an output layer and an intermediate or "hidden" layer, as illustrated in Fig. 2. Information flows through the network from bottom to top. First the letters units at the base are clamped, then the states of the hidden units are determined by Eqs. 1 & 2, and finally, the states of the phoneme units at the top are determined (30).

Representations of Letters and Phonemes

There are seven groups of units in the input layer, and one group of units in each of the other two layers. Each input group encodes one letter of the input text, so that strings of seven letters are presented to the input units at any one time. The desired output of the network is the correct phoneme, or contrastive speech sound, associated with the center, or fourth, letter of this seven letter "window". The other six letters (three on either side of the center letter) provide a partial context for this decision. The test is stepped through the window letter-by-letter. At each step, the network computes a phoneme, and after each word the weights are adjusted according to how closely the computed pronunciation matches the correct one.

The letters and phonemes are represented in different ways. The letters are represented locally within

each group by 26 dedicated units, one for each letter of the alphabet, plus an additional 3 units to encode punctuation and word boundaries. The phonemes, in contrast, are represented in terms of 23 articulatory features, such as point of articulation, voicing, vowel height, and so on, as summarized in Table 1. Three additional units encode stress and syllable boundaries. This is a distributed representation since each output unit participates in the encoding of several phonemes (18).

The hidden units neither receive direct input nor have direct output, but are used by the network to form internal representations that are appropriate for solving the mapping problem of letters to phonemes. The goal of the learning algorithm is to adjust the weights between the units in the network in order to make the hidden units good feature detectors.

Learning Algorithm

Two texts were used to train the network: Phonetic transcriptions from informal, continuous speech of a child (19) and a 20,012 word corpus from a dictionary (20). A subset of 1000 words was chosen from this dictionary taken from the Brown corpus of the most common words in English (21). The corresponding letters and phonemes were aligned and a special symbol for continuation, "-", was inserted whenever a letter was silent or part of a graphemic letter combination, as in the conversion from "phone" to the phonemes /f-on-/ (see Table 1). Two procedures were used to move the text through the window of 7 input groups. For the corpus of informal, continuous speech the text was moved through continuously with word boundary symbols between the words. Several words or word fragments could be within the window at the same time. For the dictionary, the word were placed in random order and were moved through the window individually.

The weights were incrementally adjusted during the training according to the discrepancy between the desired and actual values of the output units. For each phoneme, this error was "back-propagated" from the output to the input layer using the learning algorithm introduced by Rumelhart, et al. (17). Each weight in the network is adjusted to minimize its contribution to the total mean square error between the desired and actual outputs. Briefly, the weights were updated according to:

$$w_{ij}^{(n)}(t+1) = \alpha w_{ij}^{(n)}(t) + (1-\alpha)\varepsilon\delta_i^{(n+1)}P_j^{(n)} \quad (3)$$

where $w_{ij}^{(n)}$ is the weight from the j*th* unit in layer n to the i*th* unit in layer n + 1, the parameter α smooths the gradient by over-relaxation (typically 0.9), ε controls the rate of learning (typically 2.0). The error signal $\delta_i^{(n)}$ for layer n was calculated starting from the output layer N:

$$\delta_i^{(N)} = (p_i^* - p_i^{(N)})P'(E_i^{(N)}) \quad (4)$$

and recursively back-propagating the differences to lower layers

$$\delta_i^{(n)} = \sum_j \delta_j^{(n+1)} w_{ji}^{(n)} P'(E_i^{(n)}),$$

where $P'(E)$ is the first derivative of $P(E)$, p_i^* was the desired value of the i*th* unit in the output layer, and $p_i^{(N)}$ was the actual value obtained from the network. For most of the simulations the error signal was back-propagated only when the difference between the actual and desired values were greater than a margin of 0.1. The gradients in Eq. 4 were accumulated over several letters and Eq. 3 was applied only once for each word. The weights in the network were always initialized to small random values uniformly distributed between −0.3 and 0.3; this was necessary to differentiate the hidden units.

Performance

A simulator was written in the C programming language for configuring a network with arbitrary connectivity, training it on a corpus and collecting statistics on its performance. A network of 10,000 weights had a throughput during learning of about 2 letters/sec on a VAX 780 FPA. Two measures of performance were computed. The output was considered a "perfect match" if the value of each articulatory feature was within a margin of 0.1 of its correct value. This was a much stricter criterion than the "best guess", which was the phoneme making the smallest angle with the output vector. The performance was also assayed by "playing" the output string of phonemes and stresses through DECtalk, bypassing the front end that converts letters to phonemes.

Continuous Informal Speech

This corpus was a difficult one because the same word was often pronounced several different ways; phonemes were commonly modified or elided at word boundaries. The learning curve for 1024 words from the informal speech corpus is shown in Fig. 3. The percentage of correct best guesses for the phonemes rose rapidly at first and continued to rise at slower rate throughout the learning, reaching 95% after 50,000 words. Perfect matches were rarer, but were at 55% and still rising at the termination of the simulation. Primary and secondary stresses and syllable boundaries

Table 1 Articulatory representation of phonemes and punctuations[a]

Symbol	Phoneme	Articulatory features
/a/	f*a*ther	Low, Tensed, Central2
/b/	*b*et	Voiced, Labial, Stop
/c/	b*ough*t	Unvoiced, Velar, Medium
/d/	*d*ebt	Voiced, Alveolar, Stop
/e/	b*a*ke	Medium, Tensed, Front2
/f/	*f*in	Unvoiced, Labial, Fricative
/g/	*g*uess	Voiced, Velar, Stop
/h/	*h*ead	Unvoiced, Glottal, Glide
/i/	P*e*te	High, Tensed, Front1
/k/	*K*en	Unvoiced, Velar, Stop
/l/	*l*et	Voiced, Dental, Liquid
/m/	*m*et	Voiced, Labial, Nasal
/n/	*n*et	Voiced, Alveolar, Nasal
/o/	b*oa*t	Medium, Tensed, Back2
/p/	*p*et	Unvoiced, Labial, Stop
/r/	*r*ed	Voiced, Palatal, Liquid
/s/	*s*it	Unvoiced, Alveolar, Fricative
/t/	*t*est	Unvoiced, Alveolar, Stop
/u/	l*u*te	High, Tensed, Back2
/v/	*v*est	Voiced, Labial, Fricative
/w/	*w*et	Voiced, Labial, Glide
/x/	*a*bout	Medium, Central2
/y/	*y*et	Voiced, Palatal, Glide
/z/	*z*oo	Voiced, Alveolar, Fricative
/A/	b*i*te	Medium, Tensed, Front2 + Central1
/C/	*ch*in	Unvoiced, Palatal, Affricative
/D/	*th*is	Voiced, Dental, Fricative
/E/	b*e*t	Medium, Front1 + Front2
/G/	sin*g*	Voiced, Velar, Nasal
/I/	b*i*t	High, Front1
/J/	*g*in	Voiced, Velar, Nasal
/K/	se*x*ual	Unvoiced, Palatal, Fricative + Velar, Affricative (Compound: [k] + [S])
/L/	bott*le*	Voiced, Alveolar, Liquid
/M/	absy*m*	Voiced, Dental, Nasal
/N/	butto*n*	Voiced, Palatal, Nasal
/O/	b*oy*	Medium, Tensed, Central1 + Central2
/Q/	*qu*est	Voiced, Labial + Velar, Affricative, Stop
/R/	b*ir*d	Voiced, Velar, Liquid
/S/	*sh*in	Unvoiced, Palatal, Fricative
/T/	*th*in	Unvoiced, Dental, Fricative
/U/	b*oo*k	High, Back1
/W/	b*ou*t	High + Medium, Tensed, Central2 + Back1
/X/	e*x*cess	Unvoiced, Affricative, Front2 + Central1
/Y/	c*u*te	High, Tensed, Front1 + Front2 + Central1
/Z/	lei*s*ure	Voiced, Palatal, Fricative
/@/	b*a*t	Low, Front2
/!/	Nazi	Unvoiced, Labial + Dental, Affricative (Compound: [t] + [s])
/#/	examine	Voiced, Palatal + Velar, Affricative (Compound: [g] + [z])
/*/	*o*ne	Voiced, Glide, Front1 + Low, Central1 (Compound: [w] + [^])
/:/	logi*c*	High, Front1 + Front2
/^/	b*u*t	Low, Central1
/-/	Continuation	Silent, Elide
/–/	Word Boundary	Pause, Elide
/./	Period	Pause, Full Stop

a. Output representations for phonemes and punctuations. The symbols for phonemes in square brackets are a superset of ARPAbet (4) and are associated with the sound of the italicized part of the adjacent word. Compound phonemes were introduced when a single letter was associated with more than one primary phoneme. The continuation symbol was used when a letter was silent. Two or more of the following 17 articulatory feature units were used to represent each phoneme: *Position in mouth*: Labial = Front1, Dental = Front2, Alveolar = Central1, Palatal = Central2, Velar = Back1, Glottal = Back2; *Phoneme Type*: Stop, Nasal, Fricative, Affricative, Glide, Liquid, Voiced, Tensed; *Vowel Frequency*: High, Medium, Low. Four additional output units were used to represent each punctuation: Silent, Elide, Pause, Full Stop.

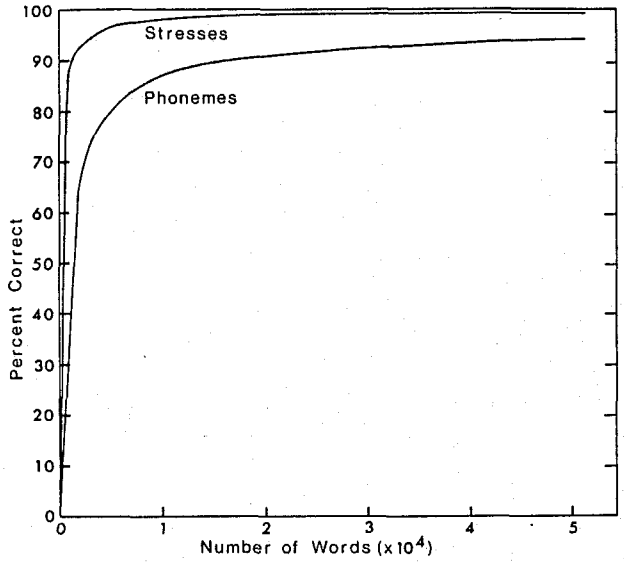


Figure 3 Learning curves for phonemes and stresses during training on the 1024 word corpus of continuous informal speech. The percent of correct best guesses are shown as functions of the number of training words.

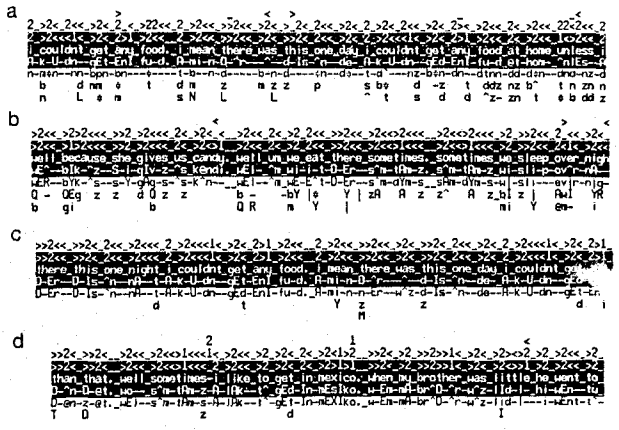


Figure 4 Examples of raw output from the simulator during learning on a corpus of 1024 words of continuous informal speech: (a) after the first 200 words of training starting from random weights, (b) after one pass through the corpus, and (c) after 25 passes through the corpus. (d) Output of the network during testing on a continuation of the corpus. The letters within the black stripe are the text (middle row), phonemes (bottom row) and stresses (top row) from the training corpus. The symbols for the phonemes are given in Table 1. The stresses are represented by a number (primary = 1, secondary = 0, tertiary = 2), and the syllable boundaries are indicated by a reversal in the direction of the arrows: "< >". The output of the network is shown above and below the black stripe. The phonemes making the smallest angle with the output vector are shown in rank order below the black stripe with the best guess at the top. The stresses are similarly listed in rank order above the black stripe with the best guess at the bottom.

were learned very quickly for all words and achieved nearly perfect performance by 5,000 words, as shown in Fig. 3.

Representative examples of output at the beginning and near the end of the training are shown in Fig. 4. The distinction between vowels and consonants was made early; however, the network substituted the same vowel for all vowels and the same consonant for all consonants, which resulted in a babbling sound. A second stage occured when word boundaries are recognized, and the output then resembled pseudowords. After just a few passes through the network many of the words were intelligible, and by 10 passes the text was understandable.

Errors in the best guesses were far from random. For example, few errors in a well-trained network were confusions between vowels and consonants: most confusions were between phonemes that were very similar, such as the difference in voicing between the "th" sounds in "thesis" and "these". Some errors were due to inconsistencies in the original training corpus. Nevertheless, the intelligibility of the speech was quite good.

A network trained on a 1024 word corpus of informal speech was tested without training on a 439 word continuation from the same speaker. The performance was 78% best guesses and 35% perfect matches, which indicates that much of the learning was transferred to novel words. An excerpt from the new corpus is shown in Fig. 4d.

A graphical summary of the weights between the letter units and some of the hidden units is shown in Fig. 5. The pattern of excitatory and inhibitory weights to a hidden unit can be considered its "receptive field", in analogy with the receptive fields of sensory neurons. Most hidden units responded to more than one pattern of letters. We examined performance of a highly-trained network to random perturbations of the weights. As shown in Fig. 6, random perturbations of the weights uniformly distributed on the interval $[-0.5, 0.5]$ had little effect on the performance of the network, and degradation was gradual with increasing damage. Since the distribution of the weights had a standard deviation of 1.2, the amount of information conveyed by each weight is only a few bits. Relearning was about ten times faster than the original learning starting from the same level of performance. Similar fault tolerance and fast recovery from damage has also been observed using the Boltzmann learning algorithm (15).

Dictionary

We used the 1000 most common English words to study how the performance of the network and learn-

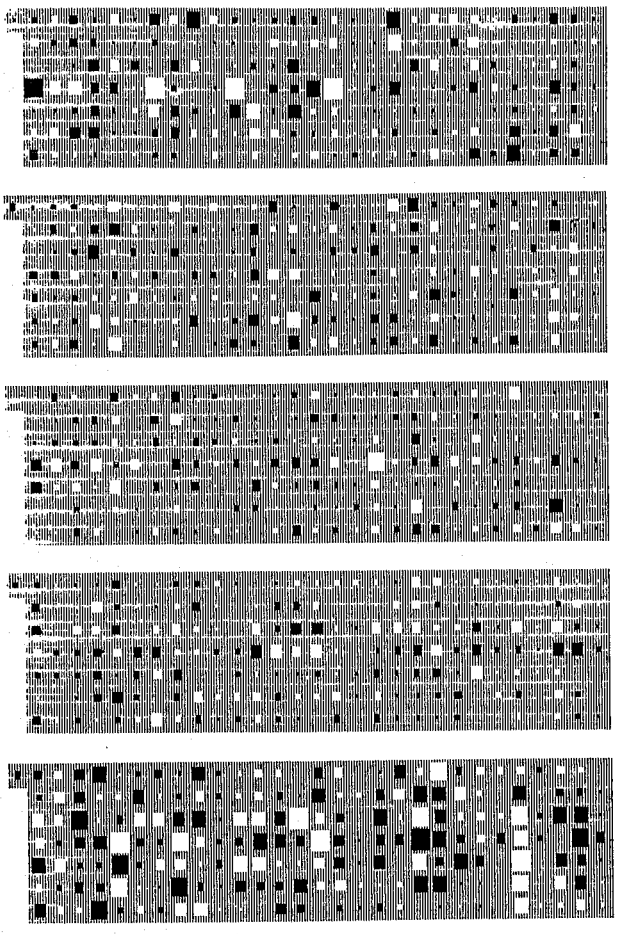
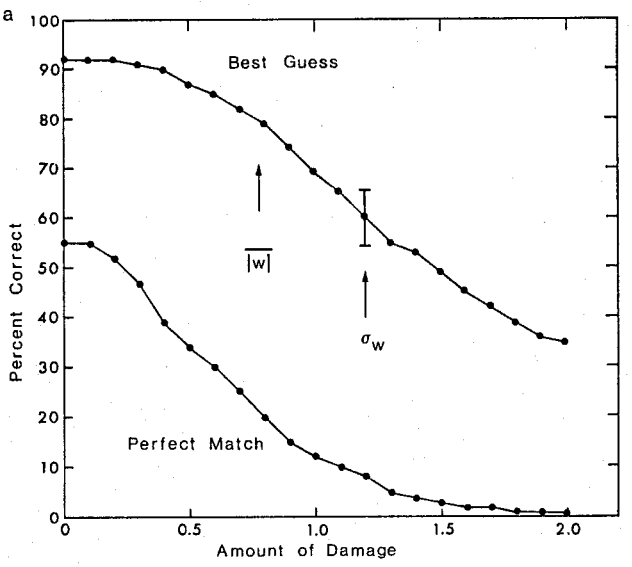


Figure 5 Hinton diagram showing weights from the layer of input units to 5 representative hidden units taken from a network with 80 hidden units that was trained on a corpus of continuous informal speech. Each gray rectangle represents one hidden unit and each square within a rectangle represents a weight. The area of the square is proportional to the magnitude of the weight and the sign of the weight is indicated by the color: white for positive, or excitatory weights, and black for negative, or inhibitory weights (28). The largest weights have magnitudes of about 4. Each row of squares within a gray rectangle represents the weights from one group of input units, with the leftmost input group on the top row and the rightmost input group on the bottom row. The isolated weight in the upper left corner of each array is the bias (negative threshold) of the unit, which was treated like a weight to a true unit. Out of 29 squares in each row, the first 26 represent the weights from the letters of the alphabet, from *a* to *z*, and the last three represent the punctuations, including word boundaries. Thus, the square in the lower left corner of a hidden unit is the weight it receives when the letter "a" is present in the rightmost input group. For most hidden units more than one combination of letters will cause it to produce a large output. These are called distributed representations. However, a few hidden units, such as the third from the top, had a more restricted pattern of weights that could be called a local representation.



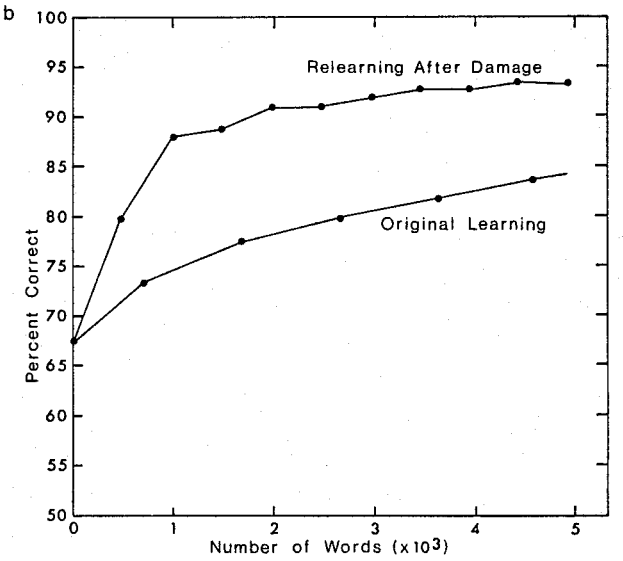


Figure 6 Damage to the network and recovery from damage. a) Performance of a network as a function of the amount of damage to the weights. The network had been previously trained on 50 passes though the corpus of continuous informal speech. The weights were then damaged by adding a random component to each weight uniformly distributed on the interval $[-d, d]$, where d is the amount of damage plotted on the abscissa. The performance shown is the average of at least two disrupted networks for each value of d. For $d = 1.2$, 22 disrupted networks were tested to obtain a standard deviation of 6%. The average absolute value of the weights in the network was $|w| = 0.77$ and the standard deviation was $\sigma = 1.2$, as indicated by the arrows. The best guesses were more resistant to damage than the perfect matches. There was little degradation of the best guesses until $d = 0.5$, and the falloff with increasing damage was gentle. (b) Retraining of a damaged network compared with the original learning curve starting from the same level of performance. The network was damaged with $d = 1.2$ and was retrained using the same corpus and learning parameters that were used to train it. There is a rapid recovery phase during the first pass through the network followed by a slower healing process similar in time course to the later stages of the original training. These two phases can be accounted for by the shape of the error metric in weight space, which typically has deep ravines (15).

ing rate scale with the number of hidden units. The most common English words are also amongst the most irregular, so this was also a test of the capacity of the network to absorb exceptions. With no hidden units, the performance rose quickly and saturated at 82% best guesses as shown in Fig. 7a. This represents the part of the mapping that can be accomplished by linearly separable partitioning of the input space (13). The pattern of errors was different from that observed in networks with a layer of hidden units in that many were stereotyped and inappropriate. Hidden units allow more contextual influence by recognizing higher-order features amongst combinations of input units (14).

The rate of learning and final performance increased with the number of hidden units, as shown in Fig. 7a. The best performance achieved with 120 hidden units was 98% best guesses, better than the performance achieved with continuous informal speech, which was more difficult because of the variability in real-world speech. Examples of two letter-to-sound correspondences are shown in Fig. 7b. The network with 120 hidden units was tested on the randomized dictionary of 20,012 words. Without learning, the average performance was 77% best guesses and 28% perfect matches. Following 5 training passes through the dictionary, the performance increased to 90% best guesses and 48% perfect matches.

Letters and punctuations were represented by single units in the input groups; this is a local representation and had the advantage that the receptive fields of the hidden units were more easily interpreted in terms of letters. Simulations were also performed with distributed representations, similar in spirit to the articulatory representation used for the output units. For a particular distributed representation with 16 units per input group we found that the general level of performance was comparable to that with the local representation, but there was a consistent difficulty with several correspondences between letters and phonemes.

We used the learning algorithm to discover a good distributed input representation by introducing an additional group of 10 units between each input letter group and the group of hidden units. The resulting network had three layers of modifiable weights. The performance of this network with 160 hidden units in the layer after training on 5 passes through the 20,012 word dictionary was 89% best guesses and 49% perfect matches. The number of weights in this network was comparable to a network with a local input representation and 76 hidden units. None of the difficulties experienced with previous hand-crafted distributed input representations occurred. As expected,

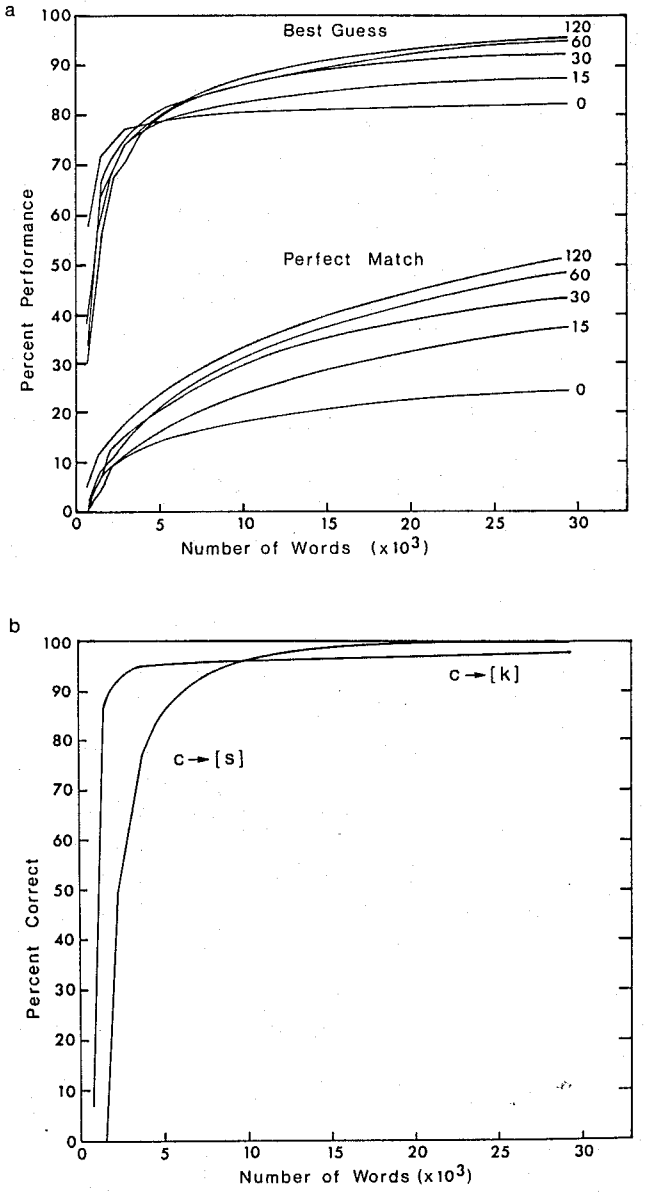


Figure 7 (a) Learning curves for training on a corpus of the 1000 most common words in English using different numbers of hidden units, as indicated beside each curve. For the case with no hidden units, the input units were directly connected to the output units. Both the percent correct best guesses and perfect matches are shown. (b) Performance during learning of two representative phonological rules, the hard and soft pronunciation of the letter "c". Note that the soft "c" takes longer to learn, but eventually achieves perfect accuracy. The hard "c" occurs about twice as often as the soft "c" in the training corpus. Children show a similar difficulty with learning to read words with the soft "c" (29).

each group of 10 units developed a highly distributed representation.

Computational Complexity

The translation of letters to phonemes can be analyzed as a mapping problem. Consider a domain of 29 symbols for letters and punctuations taken in strings of length 7. We would like to construct a deterministic mapping from these strings to a range of 51 symbols representing phonemes (23). Only a subset of all possible mappings actually occur in English speech and the problem is to find a compact description of this mapping which takes advantage of the regularities and also captures the exceptions (24).

For a restricted text this problem can be solved by specifying entries in a look-up table determined by letter strings in English words. For a text of 1000 words this would consist of about 5,000 entries since there are 5 letters on average per word and there would be at most one entry in the table per letter. However, this look-up table generalizes poorly when applied to new words in an unrestricted text. One way to generalize is to look for partial matches; this could be implemented by compiling frequency tables of letter pairs, triples, etc. for all combinations of positions within the window. There are two practical problems with this method: First the size of these tables grows exponentially with the size of the window, with about 500,000 entries needed for a text of 1000 words and a window of 7 letters. Second, some weighting scheme is required to combine evidence from different partial matches.

In NETtalk, the weighting of input letters is performed by the weights between the letter units and the hidden units, and the weighting of the features is performed by the weights between the hidden units and the output units. The learning algorithm discovers those combinations of letters that are particularly efficient at implementing the correspondences between letters and phonemes (14). The mapping is distributed in that each significant combination of letters is encoded by several hidden units, and each hidden unit recognizes more than one sequence of letters; as a consequence, the performance of the network is highly resistant to both localized and diffuse damage. Exceptions to regularities are also recognized by their features so that a separate look-up table such as that used in DECtalk is unnecessary. We are currently examining assemblies of units that appear to be related to particular letter-to-sound correspondence rules. Learning algorithms make it possible to design these efficient mappings without direct human intervention.

Biological Implications

The processing units used in the network share some properties with neurons, such as a high degree of connectivity, summation of excitatory and inhibitory influences through synaptic weights, and a nonlinear input-output function that resembles the firing rate of a neuron as a function of integrated synaptic inputs, but there are also many differences, such as the absence of explicit action potentials and an integration time constant. However, insights may be gained concerning the representation of information in large populations of neurons by examining the way that these simple network models solve problems like text-to-speech. Although the detailed implementations may be different, similar principles may apply to both neural networks and massively-parallel network models (25).

During the early stages of learning in NETtalk, the sounds produced by the network are uncannily similar to early speech sounds in children (26). However, our model of text-to-speech combines two different processes that occur at different stages of human development: learning to talk and learning to read. By the time that a human child learns to read, phonetic representations for words are already well developed. Nonetheless, the phonological mappings produced by NET-talk are efficient encodings for a parallel network and may be comparable to those used by humans.

NETtalk can be used to study the importance of particular phonological rules in the context of a particular corpus by presenting the network with nonsense words that are constructed to critically test a proposed rule. The performance of the network can also be studied following damage of the network. The patterns of errors following simulated "lesions" in the network by either removing units or by disrupting the weights can be compared with reading errors observed in humans suffering from acquired dyslexia (27).

NETtalk is clearly limited in its ability to handle ambiguities that require syntactic and semantic levels of analysis. It is perhaps surprising that the network was capable of reaching a significant level of performance using a window of only seven letters. A human level of performance would require information from larger parts of sentences to control intonation, stress contours and prosody. It should be possible to incorporate these variables into a structured network and apply the learning algorithm to them as well.

References

1. N. Chomsky and M. Halle, M., *The Sound Pattern of English*, (Harper & Row, New York, 1968); R. L. Venezky, *The Structure of English Orthography*, (Mouton, The Hague, 1970); L. Henderson,

Orthography and World Recognition in Reading, (Academic Press, New York, 1982).

2. For a discussion of exceptions in knowledge representations based on inheritance hierarchies see D. W. Etherington and R. Reiter, [*Int. Joint Conf. on Artificial Intelligence*, (William Kauffman, Inc., Los Altos, California, 1983), pp. 104–108] and D. S. Touretzky, D. S. [*Proc. National Conf. Artif. Intelligence*, (William Kauffman, Inc., Los Altos, California, 1984), pp. 322–325]. An alternative network formulation of this problem based on evidence theory is given by L. Shastri, and J. A. Feldman, [*Proc. 9th International Joint Conf. on Artificial Intelligence*, (William Kauffman, Inc., Los Altos, California, 1985)].

3. W. Haas, *Phonographic Translation*, (Manchester University Press: Manchester, 1970). For an account of abduction in plausible inference, see Ch. 8 in E. Charniak and D. McDermott, *Artificial Intelligence*, (Addison Wesley, Reading, MA, 1985).

4. Digital Equipment Corporation, *DTC-01-AA* For a study of speech synthesis intelligibility, see D. B. Pisoni, H. C. Nusbaum, B. G. Greene, *Proc. IEEE* **73**, 1665 (1985).

5. J. Allen, *Proc. IEEE* **64**, 433 (1976); D. Klatt, D. *J. Acoust. Soc. Am.* **67**, 971 (1980).

6. S. R. Hertz, J. Kadin, K. J. Karplus, *Proc. IEEE* **73**, 1589, (1985).

7. D. J. Amit, H. Gutfreund, H. Sompolinsky, *Phys. Rev. A* **32**, 1007 (1985); J. A. Anderson, *IEEE Trans. Systems, Man, Cybernetics* **13**, 799 (1983); M. A. Arbib, *Annals Biomed. Eng.* **3**, 238 (1975); D. H. Ballard, G. E. Hinton, T. J. Sejnowski, *Nature* **306**, 21 (1983); A. G. Barto, R. S. Sutton and C. W. Anderson, *IEEE Trans. Systems, Man, Cybernetics* **13**, 835 (1983); M. A. Cohen and S. Grossberg, *IEEE Trans. Systems, Man, Cybernet.* **13**, 875 (1983); S. E. Fahlman, G. E. Hinton, T. J. Sejnowski, in Proceedings of the National Conference on Artificial Intelligence, (William Kauffman, Inc., Los Altos, Proceedings of the National Conference on Artificial Intelligence, 1983), pp. 109–113; G. E. Hinton and J. A. Anderson (Eds), *Parallel Models of Associative Memory*, (Erlbaum Associates, Hillsdale, NJ, 1981); J. A. Feldman, D. H. Ballard, *Cognitive Science* **6**, 205 (1982); T. Hogg and B. A. Huberman, *Proc. National Acad. Sci. USA* **81**, 6871 (1984); J. J. Hopfield, *Proc. National Acad. Sci. USA* **79**, 2554 (1982); C. Koch, J. Marroquin and A. Yuille, *Proc. National Acad. Sci. USA* (in press); T. Kohonen, *Self-Organization and Associative Memory*, (Springer-Verlag, New York, 1984); J. A. McClelland and D. E. Rumelhart, *Psych. Rev.* **88**, 375 (1981); P. Peretto, *Biological Cybernetics* **50**, 51 (1984); P. Smolensky in *Proc. of the National Conference on Artificial Intelligence*, (William Kauffman, Inc., Los Altos, California, 1983), pp. 378–382; G. Tolouse, S. Dehaene and J.-P. Changeux, *Proc. National Acad. Sci.* (in press); C. von der Malsburg and E. Bienenstock, In: *Disordered Systems and Biological Organization*, F. Fogelman, F. Weisbuch and E. Bienenstock, Eds. (Springer-Verlag, Berlin, 1986); D. K. Waltz and J. B. Pollack, *Cog. Sci.* **9**, 51 (1985); S. Wolfram, *Physica D* in press (1986).

8. G. E. Hinton, T. J. Sejnowski, *Proceedings of the IEEE Computer Society Conference on Computer Vision & Pattern Recognition*, Washington, D.C., pp. 448–453 (1983); D. H. Ackley, G. E. Hinton, T. J. Sejnowski, *Cognitive Science* **9**, 147 (1985).

9. For example, D. Marr and T. Poggio [*Science* **194**, 283 (1976)] designed a network that could compute depth from random-dot stereograms; T. J. Sejnowski and G. E. Hinton, [in *Vision, Brain and Cooperative Computation*, M. A. Arbib and A. R. Hanson (Eds.) (MIT Press, Cambridge, MA, 1986)] designed a network to separate figure from ground in images; J. J. Hopfield & D. Tank, [*Biolog. Cybernetics* **52**, 1 (1985)] designed a network that finds moderately good tours for the traveling salesman problem quickly: Their network model is a symmetric version of the continuous nonlinear model analyzed by T. J. Sejnowski in *Parallel Models of Associative Memory*, G. E. Hinton and J. A. Anderson (Eds.) (Erlbaum Associates, Hillsdale, NJ, 1981), pp. 189–212.

10. B. Selman, and G. Hirst, *Proc. 7th Annual Conf. of the Cognitive Science Soc.* (1985); M. Fanty, University of Rochester Computer Science Technical Report TR-174 (1985); see also A. S. Lapedes and R. M. Farber, *Physica D*, (in press).

11. D. E. Rumelhart and J. L. McClelland, On Learning the Past Tenses of English Verbs, in: D. E. Rumelhart and J. L. McClelland, (Eds.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* (MIT Press, Cambridge, 1986)

12. The perceptron, which was introduced by F. Rosenblatt [*Principles of Neurodynamics*, (Spartan Books, New York, 1959)], uses binary threshold units that are deterministic, but Rumelhart and McClelland use a probabilistic update rule that turns their network into a one-layer Boltzmann machine (8). It can be shown that the perceptron learning algorithm in this case is identical to the Boltzmann learning algorithm where co-occurrence statistics are only collected for one iteration.

13. Perceptrons can only learn first-order predicates. [M. Minsky & S. Papert, *Perceptrons*, (MIT Press, Cambridge, 1969)]. McClelland and Rumelhart used a third-order coding scheme to pre-process the inputs and post-process the outputs, which made it possible to learn the mapping with only one layer of modifiable weights.

14. T. J. Sejnowski, P. K. Kienker, and G. E. Hinton, *Physica D*, (in press).

15. G. E. Hinton and T. J. Sejnowski, Learning and Relearning in Boltzmann Machines in: D. E. Rumelhart and J. L. McClelland, (Eds.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* (MIT Press, Cambridge, 1986)

16. A. G. Barto and C. W. Andersen, *Proc. 7th Annual Conf. Cognitive Science Sc.* (1985); D. B. Parker, MIT Center for Computational Research in Economics and Management Science, TR-47 (1985); Y. LeCun, In: *Disordered Systems and Biological Organization*, F. Fogelman, F. Weisbuch and E. Bienenstock, Eds. (Springer-Verlag, Berlin, 1986);

17. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning Internal Representations by Error Propagation, in: D. E. Rumelhart and J. L. McClelland, (Eds.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* (MIT Press, Cambridge, 1986).

18. G. E. Hinton, J. L. McClelland, and D. E. Rumelhart, Distributed Representations, in: D. E. Rumelhart and J. L. McClelland, (Eds.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* (MIT Press, Cambridge, 1986).

19. E. C. Carterette, and M. G. Jones, *Informal Speech.* (University of California Press, Los Angeles, 1974).

20. *Merriam Webster's Pocket Dictionary, 1974.*

21. H. Kuchera and W. N. Francis, *Computational Analysis of Modern-Day American English*, (Brown University Press, Providence, RI, 1967).

22. This supervised learning algorithm only uses positive examples and requires that the teacher correct every error for every output unit. Although children are corrected while they are learning to read, some learning also occurs through non-supervised observation of correct reading. Non-supervised learning algorithms would be worth exploring in the context of this problem [S. Grossberg, *Biolog. Cybernetics* **23**, 121 (1976); E. L. Bienenstock, L. N. Cooper, and P. W. Munro, *J. Neuroscience* **2**, 32 (1982); D. E. Rumelhart and D. Zipser, *Cog. Sci.* **9**, 75 (1985)].

23. A. N. Kolmogorov [*Dokl. Akad. Nauk SSSR* **114**, 953; *AMS Translation* **2**, 55 (1957)] has studied the class of functions that can be computed with a layered network of nonlinear processing units, and these results have been extended by G. Palm [*Biol. Cybernetics* **31**, 119 (1978)]. S. Wolfram [*Nature* **311**, 419 (1984)] has studied the

computational complexity of cellular automata, a related architecture with discrete states. The complexity analysis of what can be learned as opposed to what can be computed has only recently been addressed. L. Valiant [*Communications of the ACM*, **27**, 1134 (1984); *Proc. 9th International Joint Conf. on Artifical Intelligence*, (William Kauffman, Inc., Los Altos, California, 1985) pp. 560–566] has analyzed the learning of disjunctions of conjunctons and found a subclass that can be learned in polynomial time. See also E. M. Gold, *Inform & Control* **16**, 447 (1967).

24. A Wijk [in: *Alphabets for English*, W. Haas (Ed.), (Manchester University Press, Manchester, England, 1969)] has estimated that there are 102 graphemes or basic letter groupings in English, and P. R. Hanna, J. S. Hanna, R. E. Hodges, R. E. and E. H. Rudorf, [in: *Phoneme-Grapheme Correspondences as Cues to Spelling Improvement*, (US Department of Health, Education and Welfare, Washington, D.C., 1966)] used 170 graphemic patterns in designing phoneme-grapheme correspondence rules for spelling. However, additional rules are also needed to specify the segmentation of unrestricted English text, which is an unsolved problem. Henderson (1, p. 82) states that "... in the absence of morphological constraints it seems clear that no segmenting procedure can be formulated so as to result in a correct translation of all English words."

25. T. J. Sejnowski, Open Questions About Computation in Cerebral Cortex, In: D. E. Rumelhart, and J. L. McClelland, (Eds.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* (MIT Press, Cambridge, 1986); D. H. Ballard, *Behavioral and Brain Sciences* (in press); D. Tank and J. J. Hopfield, *Science* (in press).

26. Rumelhart and McClelland (11) have observed stages during the network learning of past tenses of English verbs that resemble the learning of past tenses by children. These developmental patterns may be a general property of incremental learning in networks with distributed representations.

27. T. Shallice, E. K. Warrington, and R. McCarthy, *Quarterly Journal of Experimental Psychology*, **35A**, 111 (1983); A. Caramazza, G. Miceli, M. C. Silveri, and A. Laudanna, *Cognitive Neuropsychol.* **2**, 81 (1985).

28. This visual representation of the weights is much more efficient than a traditional wiring diagram. For a more extensive description of this representation of a network, see (8, 14).

29. R. L. Venezky and D. Johnson, *J. Educ. Psychol.* **64**, 109 (1973).

30. Note that the output of the network only depends on letters and not on neighboring phonemes. As a consequence, the network cannot take advantage of phonotactic regularities.

31. We are grateful to Drs. Alfonso Caramazza, Stephen Hanson, James McClelland, Geoffrey Hinton, George Miller, David Rumelhart, Timothy Shallice, and Stephen Wolfram for useful insights and helpful discussions on language and learning. Bell Communications Research at Morristown, N.J. provided computational support. We also wish to thank Drs. Peter Brown, Edward Carterette, Howard Nusbaum and Alexander Weibel for their help in obtaining corpora. C.R.R. was supported by a grant to Dr. Michael Gazzaniga at the Division of Cognitive Neuroscience at the Cornell Medical Center, and T.J.S. was supported by grants from the National Science Foundation, System Development Foundation, Sloan Foundation, General Electric Corporation, Exxon Education Foundation, Allied Corporation Foundation, Westinghouse, and Smith, Kline & French Laboratories.