LEGIBILITY NOTICE

A major purpose of the Technical Information Center is to provide the broadest dissemination possible of information contained in DOE's Research and Development Reports to business, industry, the academic community, and federal, state and local governments.

Although a small portion of this report is not reproducible, it is being made available to expedite the availability of information on the research discussed herein.

LA-UR -88-1849

TITLE: HIERTALKER: A DEFAULT HIERARCHY OF HIGH ORDER NEURAL NETWORKS THAT LEARNS TO READ ENGLISH ALOUD

LA-UR--88-1849

DE88 014455

AUTHOR(S): Z. G. An, NYNEX and CNLS
S. M. Mniszewski, C-10
Y. C. Lee, CNLS
G. Papcun, C-10
G. D. Doolen, ADDRA

SUBMITTED TO IEEE Annual International Conference on Neural Networks, July 25, 1988, San Diego, California

DISCLAIMER

Los Alamos National Laboratory
Los Alamos, New Mexico 87545

FORM NO 836 R4
ST NO 2629 5/81

HIERTALKER: A DEFAULT HIERARCHY OF HIGH ORDER NEURAL NETWORKS THAT LEARNS TO READ ENGLISH ALOUD

Z.G. An* , S.M. Mniszewski, Y.C. Lee
G. Papcun, and G.D. Doolen
Center for Nonlinear Studies
Los Alamos National Laboratory

Abstract

A new learning algorithm based on a default hierarchy of high order neural networks has been developed that is able to generalize as well as handle exceptions. It learns the "building blocks" or clusters of symbols in a stream that appear repeatedly and convey certain messages. The default hierarchy prevents a combinatoric explosion of rules. A simulator of such a hierarchy, HIERtalker, has been applied to the conversion of English words to phonemes. Achieved accuracy is 99% for trained words and ranges from 76% to 96% for sets of new words.

1. Introduction. It is a problem of general interest to determine the relationships among two or more sets of discrete symbols. This problem occurs in translating languages (both computer languages such as Pascal and Fortran, and natural human languages such as English and Chinese), determining the correspondence between genetic codes and their structural realizations, and in mapping ordinary spelling onto a phonetic transcription appropriate to drive a speech synthesizer.

This last problem has recently been studied by Sejnowski and Rosenberg[1] using a back-propagation neural network approach.[2] This system, known as NETtalk, can recall the correct pronounciation of a corpus of training words and generalize to novel words. NETtalk is based on an automated learning procedure, in contrast to traditional AI approaches (for example, DECtalk, a commercial product that converts text to speech) which rely on highly labor intensive entries of phonological rules. Additionally, Stanfil and Waltz also tackled this problem of text to phoneme translation using a memory-based reasoning approach (MBRtalk) which works directly from a data base of training set[3].

In this paper we present a new learning procedure, based on a default hierarchy of high order neural networks, which exhibits an enhanced capability of generalization and good efficiency. This new architecture is suitable for learning the regularities or "building blocks" embedded in a stream of information with inherent long range correlations. Moreover, it is not plagued by a combinatoric explosion of rules in learning. A simulator which applies this learning paradigm to the conversion of English words to phonemes was developed and is known as HIERtalker. We will show results using HIERtalker, discuss their implications, and talk about some future directions for research.

2. The Default Hierarchy. In this section we describe the default hierarchy in concept and functionality. This type of learning procedure will produce contexual rules for mapping an input stream of information to an output stream. Both streams must be aligned in some consistent fashion allowing a one-to-one mapping between items in the streams.

* Present Adress:NYNEX Science and Technology, 500 Westchester Ave ,White Plains,NY 10604

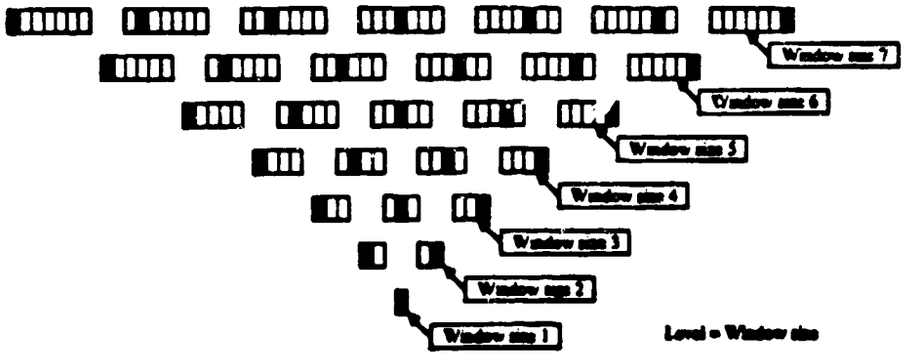

Figure 1.

A contextual rule focuses on one input item either in null context or in the context of surrounding input items to determine the appropriate output item.

Rules are partitioned in a hierarchy as shown in Figure 1. Rules at level 1 use a context window size of one, rules at level 2 use a context window size of two, rules at level 3 use a context window size of three, etc. Within a level L there are L possible context window orientations possible. For example, at level 3, one window focuses on the first or left-most of the three items and determines the output item corresponding to that first item, the second window focuses on the second or middle of the three items and determines the output item corresponding to that middle item, and the third window focuses on the last or right-most of the three items and determines the output item corresponding to that last item. All context window orientations may not be required for an application. The choice of the orientations used, as well as the number of levels required, is dependent on the characteristics of the data.

Each of these context window orientations corresponds to a high order correlation neural network [4] of order L for level L. The L context window items from a level L rule are correlated by multiplication or concatenation, depending on whether the input stream is numerical or symbolic. A symbolic rule cannot fire unless all context window items are present together. In particular, the patterns produced by the window of L letters are used as inputs to train a neural network of L-th order. We denote the input string of letters by $I_i = (I_{i1}, I_{i2}, ..., I_{in})$, and the output phoneme by O_j. The mean square error is

$$< E^2 > = < (O - O')^2 >$$

where $O = WI$, W is a matrix to be determined, O' is the desired answer, and <> indicates the ensemble average. To minimize the error we must have

$$W = < O'I > < II >^{-1}$$

where

$$< I_i I_{i'} > = < I_{i1} I_{i2} .... I_{in} I_{i1'} I_{i2'} ..... I_{in'} > .$$

Taking approximately $< II > = 1$, we have $W = < O'I >$. This is the Hebbian learning rule. Obviously, in order to optimize, the Hebbian rule is more appropriate for high order networks than for first order ones, because for high order network $< II > = 1$ is a better approximation.

The rules, "building blocks", or connections are created dynamically during the training process based on the training data. Starting at level 1 with a context window of size one,

we go through the entire training set collecting all the existing one item correspondences and the number of times each of these correspondences occurs. The most common output item becomes the rule consequent. Then we procede to learn rules at level 2. The different context window orientations are learned separately. Once again, the order in which they are most appropriately learned is once again determined by the characteristics of the application. To learn rules for the context window orientation of two items which focuses on the first item we iterate through the training set once again looking at everything through a context window of two. Before creating any rules for this context window orientation, a check is made to see if rules exist for previously trained windows (in this case level 1) that already have captured this information. Here we collect all the two item correspondences with a focus on the first item that have not been captured in previously generated rules. Once again frequency counts are collected and the most common output item becomes the consequent for a rule. This type of learning procedure is continued for each context window orientation at each level. The number of levels is variable and dependent on the complexity of the data. Rules will not be created for larger window sizes if not needed.

General rules are created with small window sizes while exception rules are created with larger window sizes. This capability of generalization of a default hierarchy stems from its logical structure and is independent of details.

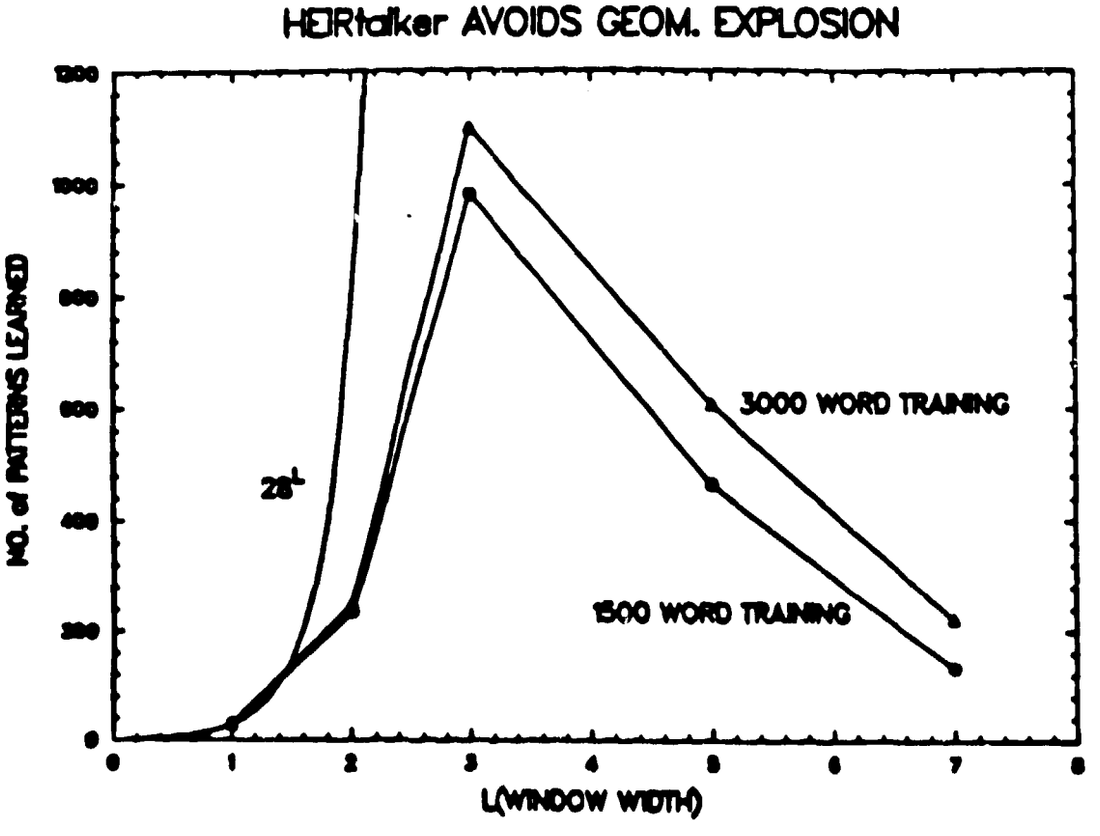

Figure 2.

The "default" aspect of the default hierarchy concept becomes more apparent in testing. Once we have trained this network, we have a hierarchy of high order correlation neural networks. Each high order correlation network corresponds to a set of rules for a particular context window orientation. We have trained these context window orientations in some chosen order from level 1 to level L. Testing is done in the reverse order. Given a stream of input items, we want to predict the corresponding stream of output items. Testing starts with the focus on the first or left-most item of the input stream. The system first searches the lowest level, L, for an appropriate rule. The search through the different context window orientations occurs in the opposite order than it was trained in. If there is a match, then an answer at this level of specification is obtained, otherwise the system defaults to the next level, L-1, to seek a less specified answer. Rules at different levels and context window orientations will be used to predict the entire corresponding output stream. A guess will always be available from level 1 if no other appropriate rules exist.

An outstanding feature of the default hierarchy is that it is not plagued with the difficulty of combinatoric explosion. This latter point is illustrated in Figure 2., where the number of indispensable rules n(L) learned in each level L is plotted as a function of L. The reason that the number n(L) decreases so rapidly with L is that only a few exceptions are learned in very low levels.

A simple mathematical argument will show why capturing the "building blocks" brings about tremendous saving. Suppose we are looking at strings of L items as our context windows. The total number of combinations in such a window is m^L, where m is the number of different input items. When L is large, m^L is too large a number to handle, and this characterizes the so called combinatoric explosion. However, if we can capture the "building blocks", and assuming all the "building blocks" are of length less than l, $(l < L)$, then the total number of patterns in a context window of length L is less than

$$L\sum_{i=1}^{l} n_i \leq L\sum_{i=1}^{l} m^i \leq Lm^{l+1}$$

where $n_i \leq m^i$ is the number of possible "building blocks" of length i $(1 \leq i \leq l)$. Therefore the factor of saving due to capturing the "building blocks" is (m^L/Lm^{l+1}), which can be large.

3. Text to Phoneme Translation. The default hierarchy, as described above, was applied to the problem of reading English words by translating the words to their appropriate phoneme representations. We mapped words (consisting of combinations from the 26 letters of the alphabet) to their phonemes (consisting of combinations of phonemes from Table 1.). The phoneme set that was chosen was based on the one letter phoneme translations used by Digital Equipment's DECtalk speech synthesizer, thus allowing the output strings of phonemes to be played back through DECtalk, bypassing the part of the machine that converts letters to phonemes. Stresses were not used.

The training sets and test sets contained words with their associated phoneme representations. In assembling these training and test sets the question of alignment arises. A precise and consistent alignment is important to allow general rules of text to phoneme translation to be captured and used. A lack of alignment leads to the generation of a large

number of inefficient rules.

PHONEME	SOUND	PHONEME	SOUND	PHONEME	SOUND
a	st*o*p	b	*b*et	c	to*ugh*t
d	*d*one	e	l*a*ke	f	*f*ill
g	*g*et	h	*h*int	i	se*a*t
k	*K*en	l	*l*et	m	*m*en
n	*n*et	o	co*a*t	p	*p*et
r	*r*un	s	*s*it	t	*t*est
u	l*u*te	v	*v*ase	w	*w*et
x	*a*bout	y	*y*ell	z	*z*oo
A	*b*ite	C	*ch*ill	D	*th*is
E	b*e*t	G	ri*ng*	I	s*i*t
J	*g*in	L	bott*le*	M	ran*som*
N	butt*on*	O	b*oy*	Q	*o*ne
R	b*ir*d	S	*sh*ow	T	*th*in
U	b*oo*k	W	sh*ou*t	X	bo*x*
Y	c*u*te	Z	a*z*ure	@	*c*at
^	b*u*t				

Table 1.

Some alignments are straightforward. For example, "top" ⟶ "tap" is already one-to-one. Another example, "cake" ⟶ "kek_" has a silent "e" which is shown by a "_". This does not present a problem because the alignment is correct until the final silent letter. The word "third" ⟶ "T_Rd" becomes more complicated. We associate "th" ⟶ "T_" and "ir" ⟶ "_R". The mapping "th" ⟶ "T_" could as well have been "_T". The first mapping does appear to be more natural. It is necessary to decide on the type of letter associations as shown above and then to be consistent in applying them throughout the data set. When there are errors or noise in the data set due to inconsistent alignments or erroneous phoneme translations, extra exception rules will be generated to compensate.

We collected and assembled a number of different data sets for our work. Some contained different kinds of alignment. Some contained errors in phoneme translation and alignment. One contained a very consistent alignment and translation. Most of these words were initially taken from a 250,000 word dictionary[6], which had been translated using the phoneme set as described above. A rough alignment was done programmatically. This dictionary was used as a resource to selectively obtain the phoneme translations of sets of words. Clean-up of the phoneme translation and alignment was done by hand. After Word Set # 1 was constructed and cleaned up, it was also used as source data for a neural network to perform automatic alignment. It did a reasonably good job, but some alignment errors still remained.

Word Set #1 consisted of 1017 words in alphabetical order from a book containing sets of words classified by phonetic sound.[7] The idea behind this training set was to collect a set of words which included as many different letter representations of all the different phonemes. This set did not contain all possible combinations.

Word Set #2 consisted of 1718 words, 1017 of which are from Word Set #1, with an additional 701 words randomly chosen from the dictionary. These 701 words were initially aligned using the alignment neural network and then corrected manually. This training set attempted to provide more examples to allow for more letter to phoneme combinations.

Word Set #3 consisted of 1441 words from a book that contained a more systematic word set, from simple to difficult.[8] These were left in the order they were presented in the book.

This training set attempted a systematic presentation of examples, starting with simple one syllable words, working up to larger words with suffixes such as -tion, -able, etc. A more efficient hierarchy with less rules was generated when this set was used as the training set in its original order than when words were presented alphabetically.

Word Set #4 consisted of 6219 words in alphabetical order. It contained Word Set #2 with an additional 4501 words randomly taken from the dictionary. The 4501 words were aligned using the alignment neural network and were not corrected manually. This training set basically contained a number of random words which contained a larger number of letter to phoneme combinations than the above sets. It also contained errors which brought about exception rules at the lower levels in the hierarchy.

Word Set #5 consisted of 1000 random words chosen from the original dictionary that were not in Word Set #4, but may have been similar.

Word Set #6 consisted of the 100 randomly chosen words used by Stanfil and Waltz for MBRtalk.[3] This set serves as a way to compare our results with MBRtalk.

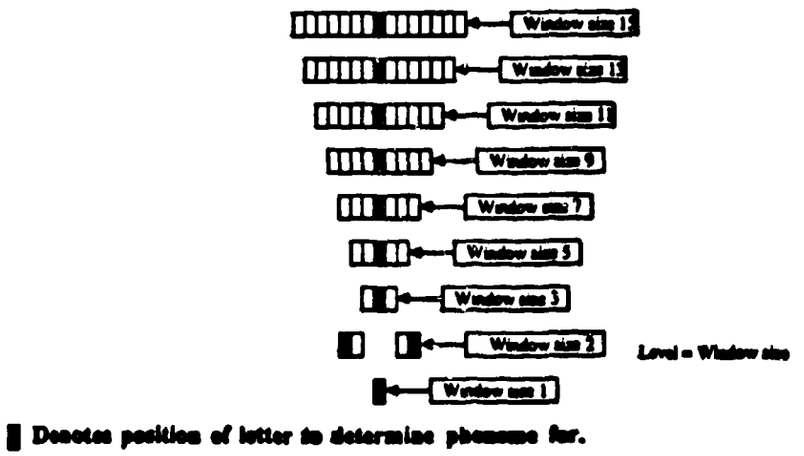

Figure 3.

The default hierarchy architecture that was used for translating words to phonemes allowed up to 15 letter windows and used mostly context window orientations which focused on the center letter as shown in Figure 3. A center orientation was chosen since it has been shown that a significant amount of the information needed to correctly pronounce a letter is contained in the surrounding letters.[9]

The original HIERtalker simulator was written in the Fortran programming language for the Cray. Due to the symbolic nature of this application, it lent itself better to a LISP implementation. A LISP version of the HIERtalker simulator was developed that runs on the Texas Instruments Explorer.

4. Results. All the performance results are based on the percentage of correct phonemes chosen by HIERtalker. The system may have come up with acceptable alternatives in some instances, but these were not counted.

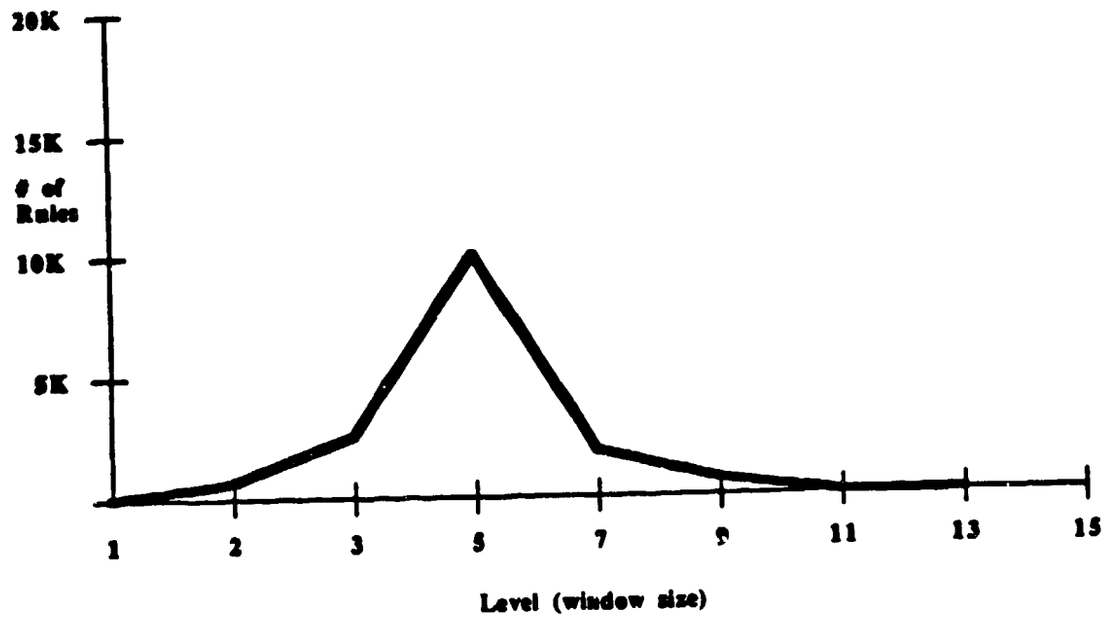

Figure 4.

The number of passes through a training set is dependent on the depth of the hierarchy and the different context window orientations used. One pass through the training data is accomplished for each context window orientation at each level.

HIERtalker was trained using Word Set #4. In Figure 4 we see that most rules were generated using a 5 letter window and the maximum size window used was 13. When tested on the training set an accuracy of 99% was achieved. The default hierarchy was able to capture the "building blocks" in the training set.

When tested on Word Set #3, an accuracy of 83% was achieved. Errors occurred for letter to phoneme combinations it had not seen before and in some cases acceptable alternatives were given.

When tested on Word Set #5 an accuracy of 96% was achieved. This high result was due to the set containing many of the "building blocks" that had been learned.

When tested on Word Set #6 an accuracy of 76% was achieved. This is lower than the 86% accuracy achieved by MBRtalk[3] but is good considering it was only trained on a set of a little over 6200 words. A number of the letter to phoneme combinations in this set had not been seen before.

HIERtalker has no problem distinguishing between vowels and consonants. It does have trouble occasionally in distinguishing between long and short vowel sounds in letter combinations it had not seen during training.

We would like to obtain some larger training sets for further study. In the future we would also like to look at the impact of additional input information such as stress, part of speech, and syllable or morpheme boundaries in determining a word's pronunciation.

5. Conclusion. We have developed a new, efficient, learning algorithm, based on the concepts of a default hierarchy and high order neural networks, and have applied it to the problem of translating English text to phonemes.

It is illustrated through this application that a default hierarchy is very efficient in learning the "building blocks" in a stream of information, which are clusters of the basic symbols of the stream that appear repeatedly, and convey consistent messages. The default hierarchy captures and utilizes these "building blocks" in a way similar to "cluster decomposition" in physics. English text happens to be one of those information streams in which there are definite "building blocks", which explains why the default hierarchy works well for it.

This algorithm generalizes readily to other important learning problems such as translating computer languages, deciphering the genetic code[5], and accelerator control. We plan to report progess on the application of the default hierachy learning algorithm to these problems in forthcoming IEEE meetings.

References

1. T.J. Sejnowski, and C.R. Rosenberg, "Parallel Networks that Learn to Pronounce English Text", *Complex Systems 1*, (1987) 145-168.
2. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation", in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations*, edited by D. E. Rumelhart and J. L. McClellend, (MIT Press, 1986) 318-362.
3. C. Stanfil and D. Waltz, "Toward Memory-based Reasoning", *Communications of the ACM*, (December, 1986) 1213-1228.
4. Y.C. Lee, G.D. Doolen, H.H. Chen, G.Z. Sun, T. Maxwell, H.T. Lee and C.L. Giles, "Machine Learning Using a Higher Order Correlation Network", *Physica 22D*, (1986) 276.
5. A. Lapedes, private communication.
6. J.E. Shoup, *American English Orthographic-Phonetic Dictionary*, Speech Communications Research Laboratory (May, 1973).
7. J. Griffith and L.E. Miner,*Phonetic Context Drillbook*, Prentice-Hall (1979).
8. A. Linksz, *On Writing, Reading, and Dyslexia*, Grune & Straton (1973).
9. J. M. Lucassen and R. L. Mercer, "An Information Theoretic Approach to the Automatic Determination of Phonemic Baseforms", *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, (1984) 42.5.1-42.5.4.