# A Comparison of the Classic NetTalk Text-to-Speech System To a Modern, Distributed Representation and Simple Recurrent Network

Shyam S. Ramamurthy
Department of Electrical Engineering

Gijun Lee, Anshul Arora, and Douglas S. Blank
Department of Computer Science and Computer Engineering

University of Arkansas, Fayetteville
{sramamu, glee, aarora, dblank}@comp.uark.edu

**Abstract**

This paper reports on a comparison to the well-known NetTalk implementation of English text-to-speech translation via neural networks. A *distributed representation scheme* for encoding is investigated opposed to the classic *localist representation scheme* used in the original NetTalk. The paper discusses a modern re-implementation based on Elman's Simple Recurrent Network.

## Introduction

The English text-to-speech translation problem attempts to implement the task of reading text outloud. This is a difficult task because spoken English is highly dependent on the context in which the words and phrases appear. Due to the context sensitivity, generalized phonological rules for transforming from words to phonetic sounds don't work. For example, the 'a' in words ending in 'ave', such as 'save', 'wave', 'brave', is a long vowel. However, this is not universal. For instance, 'have' is an exception. Also, there are words such as 'wind' that can vary pronunciation with their grammatical role. These cases can be shown by the following examples: (a) The wind is blowing. (b) I wind the clock. Such exceptions present a major obstacle to text-to-speech systems.

Hundreds of letter-to-phonetic rules are required to correctly translate 90% of the words in an unlimited textual situations (Jonathan and Klatt, 1987). The use of generalized rules becomes cumbersome since two categories must be considered: one for regular cases, and one for exceptional cases. A decision has to be then made whether a generalized rule will be used to treat a case or the specialized rule will be used to treat the irregular case.



The formulation of an appropriate set of rules for a particular set of words involves much trial and error. As an alternative to rule based translation, T.J.Sejnowski of John Hopkins University created a neural network-based translation system called NetTalk in 1986.

NetTalk uses an approach based on an automated learning procedure for a parallel network of deterministic processing units. Once trained on a vast number of speech contexts, it can process novel words producing the correct phonetic transcription. Sejnowski used a seven letter window as a preprocessing unit, this being found to be long enough to take most of the regular and exceptional cases in pronunciation into consideration. All the regular and irregular cases are mapped simultaneously into the network and treated equally.

The cost of the neural network based approach is the large amount of computation time and resources, particularly in training, when the data set is large. So this requirement for excessive computation time limited the implementation of the system. More importantly, NetTalk used a *localist representation* for encoding the input data. This encoding scheme required that each textual input (letters, puncuation, etc.) and each phonetic output be represented by its own unit in the network. As a result, the input and output vector sizes are large, thereby requiring more resources during training and implementation.

It is hence useful to know the effects of considering a *distributed representation* in the NetTalk domain. Since the process of text-to-speech conversion may be considered sequential, we propose using a Simple

Recurrent Network (SRN, Elman 1990). A SRN based implementation can reduce the input vector size further and is expected to yield a better performance than the feed-forward network implementation used by NetTalk.

In the following sections, the NetTalk system is first described. This is followed by a description of the modified implementation. The results of our experiments are then reported. Finally, the conclusions and the scope for future work is presented.

## Description of the NetTalk system

The NetTalk data set contains a list of 20,008 English words, along with a phonetic transcription for each word. The task is to train a network to produce the proper phonemes, given a string of letters as input. The data file represents each word in the form shown below for the word "abbreviate" (see Figure 1). Corresponding to each word, a pronunciation code and an accent code is listed.

| Textual Input | | Phonetic Ouput | Stress |
|---|---|---|---|
| a | => | x, | 0; |
| b | => | b, | <; |
| b | => | -, | >; |
| r | => | r, | >; |
| e | => | i, | 1; |
| v | => | v, | >; |
| i | => | i, | 0; |
| a | => | e, | 2; |
| t | => | t, | <; |
| e | => | -, | <; |

Figure 1: The word 'abbreviate' in the NetTalk data representation. Column 1 shows the textual input, column 2 the phonetic output, and column 3 the stress level for that phoneme. See text for details.

Even though a certain letter in a word might not produce a sound, NetTalk maps the alphabet to a pronunciation symbol and accent. For instance, Figure 1 shows the actual data used in NetTalk for the work "abbreviate". The first column has English letters; the second column, the actual pronunciation; and the third column, accent and intonation. For example, the first 'a' in the word abbreviate can be mute or could be pronounced like very weak 'uh,' and therefore, mapped to that phoneme represented by 'x'. The second 'b' in the word does is not pronounced because of the first 'b'. So, the second 'b' is mapped to the '-' symbol, which means that it is not uses in the pronunciation. This also applies to the last 'e'. In the third column, you can see three numbers and the less-than and greater-than symbols. NetTalk maps every vowel to one of these accent symbols. If a vowel gets a primary accent, it is marked as a two. If a vowel is marked as a secondary accent, it is represented as a one, and zero means almost no accent. For intonation, '>' and '<' are used. '>'.

The training input to the network is a series of seven consecutive letters from one of the training words. The central letter in this sequence is called the "current" one for which the phonemic output is to be produced.

Three letters on either side of this central letter provide the context that helps to determine the pronunciation. There are a many characters in the English alphabet for which this local seven-letter window is sufficient to determine the proper phonemic output. For the study using this "dictionary" corpus, individual words are moved through the window so that each letter in the word is seen in the central position. Blanks are added before and after the word as needed (see Figure 2). Some words appear more than once in the dictionary, with different pronunciations in each context; only the first pronunciation given for each word was used in this experiment.

NetTalk uses a localist encoding. For each of the seven letter positions in the input, the network has a set of 29 input units: one for each of the 26 letters in English, and three for punctuation characters. Thus, there are 29 x 5 = 203 input units in all. The output side of the network uses a distributed representation for the phonemes. There are 9 output units representing various articulatory features such as voicing and vowel height. A distinct binary vector represents each phoneme over a set of 29 units (see Figure 3).

Standard back-propagation was used, with update of the weights after the error gradient has been computed by back-propagation of errors for all the letter positions in a single word. The number of hidden units in the network was varied from one experiment to another, from 0 to 120. Each layer was totally connected to the next; in the case of 0 hidden units, the input units were directly connected to the outputs.

The network weights were initialized with random values in the range -0.3 to +0.3.

## The modified implementation

We then used a reduced data set consisting of 500 commonly used words as the basis of our experiments. This reduced data set was compared with the NetTalk data

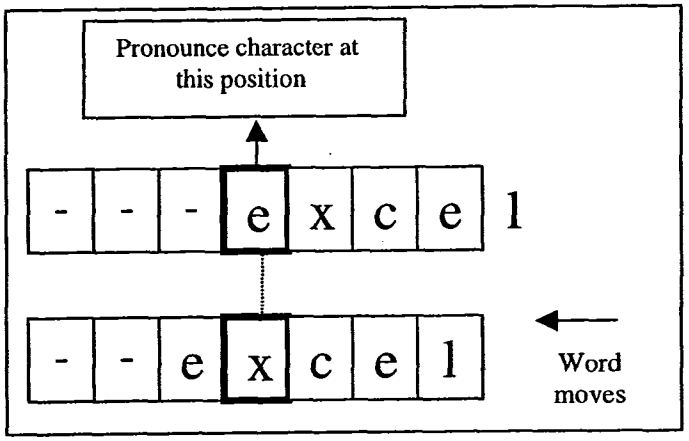


Figure 2. Illustrates the encoding of the word using the 7-character window.

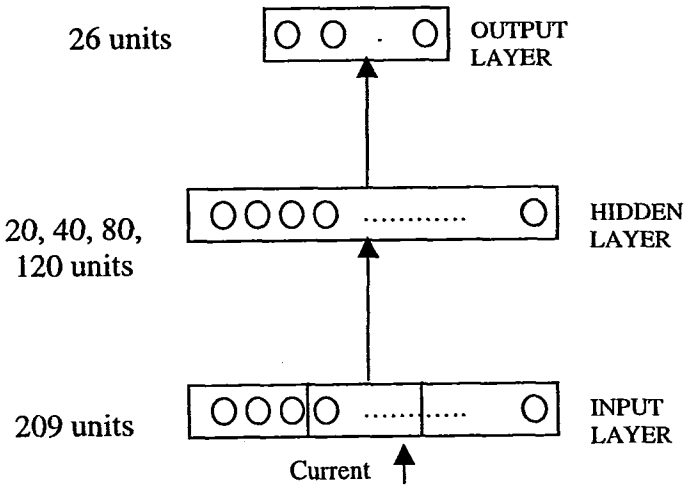


Figure 3. Feed-forward network structure used by NetTalk

file and 418 matching words were picked and used. Out of these 418 words, 400 were used for training the network and 18 randomly chosen words were used for testing the network. These 18 words result in about 50 patterns due to encoding.

Initially, we tried replicating the original NetTalk system, but with a slightly different input representation. The encoding used as input consisted of 5-bit binary numbers (thus allowing for 32 possible cases at each input) for each of the locations. We used a 7-character window as in NetTalk. Thus, we need only 5 X 7=35 units at the input. At the output, we used a 9-bit distributed encoding. Out of the 9-bits, 6 (64 cases) are for the pronunciation code and 3 (8 cases) for the accent code.

This encoding resulted in more than 1800 patterns for the data set used. First, a feed-forward neural network was implemented in the Con-x backpropagation system (Blank, 1999) with 35 units at the input layer and 9 units at the output layer. The performance of this network was tested for 20, 40 and 80 units at the hidden layer.

This encoding scheme was then tested using a SRN (Elman, 1990). In the case of the SRN, the number of units in the input layer was reduced to 20. This is because the input layer has only 4 positions consisting of the current character for which the pronunciation is to be produced and 3 leading characters. Each character requires 5 units for the distributed representation at the input (see Figure 4). The performance of this network was tested on the same data set for 20 and 40 units in the hidden layer. The number of units in the context layer was set the same as the hidden layer.

In the SRN implementation, the activations from the hidden layer were copied to the context layer whenever there were characters preceding the character in the pronunciation position. In the absence of any characters, the activations at the context layer were set to 0.5.

The ability of the SRN to discover generalizations in this problem lies in three primary components of the model. First, the sequential presentation of words encoded as binary vectors provides an reduced data set that only contains implicit information about the relationships based on the adjacency of words in the training set. Second, architectural constraints of the SRN such as the recurrent combination of each input vector with the hidden-unit activations from the previous time step provide a compressed representation of the sequence of inputs. Third, in order to solve the non-deterministic prediction task, the model is forced to form a distributed memory that is based on the similarity structure of the compressed sequence representations. It is this similarity structure in hidden unit space that reflects the lexical classes of the words (Elman 1990).

## Results

The results of the implementation using the feed-forward network and the SRN are described below. For comparison purposes, the results achieved by NetTalk on the full data-set of 20,008 words with more than 140,000 training patterns is also described.

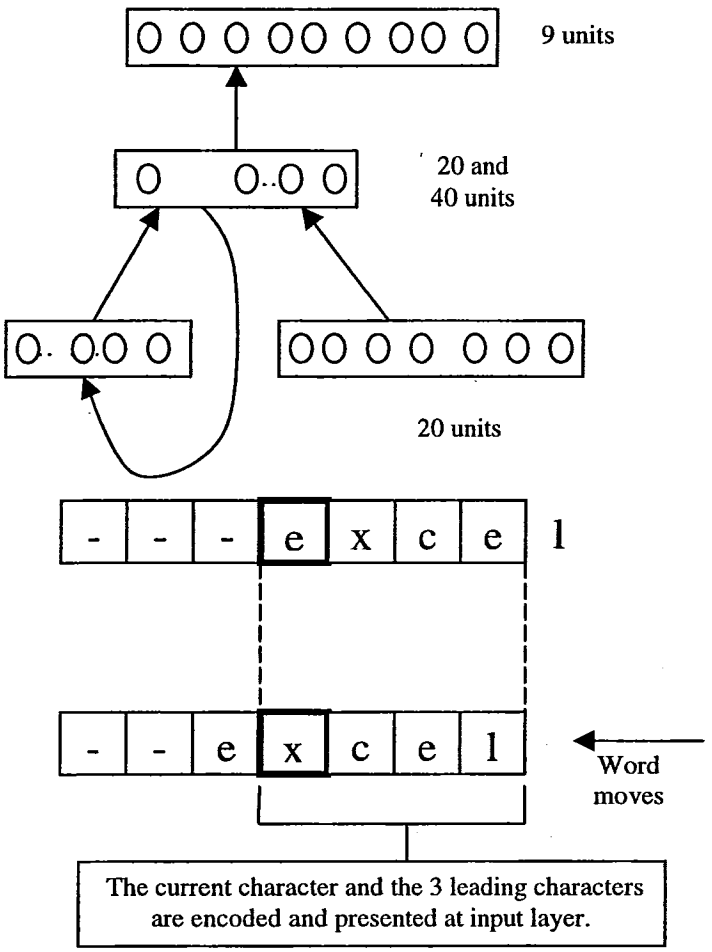


Figure 4: SRN Comparision

For the feedforward network, the original NetTalk implementation used 209 units in the input layer and 26 units in the output layer. As detailed before, the encoding was localist (orthogonal) at the input and distributed at the output. For the full data set, this resulted in an 82 % match with the test cases for 20 units in the hidden layer and a 98% match with 120 units in the hidden layer. In our feedforward network implementation, we use 35 units in the input layer and 9 units in the output layer. We have used distributed (non-orthogonal) encoding at the input as well as the output. This resulted in a 70% match with 20 units in the hidden layer and a 93% match with 40 units in the hidden layer on the reduced data set. Thus, our smaller distributed representation performed slightly worse than the original NetTalk. This can be explained (at least in part) to the reduced training set size.

The SRN implementation with 20 units in the input layer and 9 units in the output layer resulted in a 90% match with the test cases for 20 units in the hidden layer and 100% match for 40 units in the hidden layer. Again, we have used a distributed (non-orthogonal) encoding at the input as well as the output. In this case, although we used the same distributed representation as before, due to the recursive network structure we were able to out-perform the original NetTalk performance.

## Conclusions and Future work

These results indicate that it is feasible to use a distributed (non-orthogonal) representation for this problem. As a result, the number of units can be reduced leading to advantages in the implementation. The SRN based implementation indicates that this approach can be applied successfully for this application leading to possible reduction in input window size or possibility of more inputs from the leading characters. In these experiments, we have used a significantly reduced data set. The set of words used reflects the most commonly used words. The implementation has illustrated the feasibility and power of the SRN approach. Further experiments on larger data sets and with contexts of complete sentences should be done to get a better comparison and reach firm conclusions.

## References

Blank, D.S., (1999). Con-x: Connectionist Backprop Language and Simulator User's Manual. `http://dangermouse.uark.edu/cx/`

Elman, J. 1990. *Finding Structure in Time, Cognitive Science.*

Jonathan, A., and Klatt Dennis, K. 1987. *From Text To Speech: The MITalk system.* Cambridge University Press.

Sejnowski, T. J. 1986. NetTalk: A parallel network that learns to read aloud.