Text Processing For Speech Synthesis Using Parallel Distributed Models

Michael S. Scordilis
John N. Gowdy
Electrical & Computer Engineering
Clemson University
Clemson, South Carolina

Abstract

Mapping letters to phonemes is a fundamental problem in automatic speech conversion and arises from the fact that no complete set of rules exists to account for all the different pronunciations of letter groups in English. Although a number of algorithms containing thousands of rules are being used in text-to-speech synthesis, parallel distributed models have shown the potential of processing speech much more efficiently. Here, as part of a new approach to the synthesis problem, such a model is presented. This model includes a procedure which marks those incoming letters which are soundless, but which modify the pronunciation of their neighbors.

Introduction

As part of the ongoing activities aimed at improving man-machine interfaces, unrestricted speech conversion has been the subject of long research efforts. Speech synthesis is used by choice when visual information is not desired or by necessity in the case of visually impaired system users. Although several commercially available speech synthesis systems are, for the most part, successful in producing intelligible speech, the goal of synthesizing pleasant and natural sounding voices is still elusive. Renewed interest in the design of improved methods has been generated with the introduction of the CD-ROM technology [1]. Such devices are capable of storing databases in excess of 500 Mbytes.

The difficulty of the speech synthesis problem arises mainly from the nature of human speech itself. Speech is characteristically a cognitive activity, and requires an understanding of the text segment to be spoken in order for the ideal, "natural" utterance to be produced. Language understanding requires the development and use of an artificially "intelligent" machine, a difficult problem in itself. In addition to this universal property of speech production, synthesizing in English is further complicated by the anarchic nature of the English language, in which letter to sound correspondence cannot be adequately described by a simple set of rules. Word pronunciation is heavily dependent on context and is also a function of many variables a- mong which are geographical and historical factors.

Text Processing

Text-to-speech conversion begins with high level processing of the incoming text before the synthesis process is evoked. An integral part of this phase is the conversion of the symbols and abbreviations present in the text segment into standard orthographic form with the use of a set of rules and a dictionary. For example, "1989" is converted into "nineteen eighty nine" and "Mr." into "Mister" before any phonological or linguistic analysis is attempted. Once all such strings have been converted, phonetic transcription is performed on the normalized text [2]. The block diagram in figure 1 shows the typical parts of the high level processing stage in a speech synthesis system.

Phonetic transcription is the process which accepts orthographic text and produces the appropriate phonemic sequences. In other words, this is the mapping of letters into phonemes, the basic sounds of a language. Several quite successful algorithms have been used for transcribing text into phonemic strings. These algorithms contain hundreds of rules and extensive tables of exceptions and have proven quite successful [2,3,4,5]. However, all these rule-based systems recognize the fact that no complete set of rules able to cover all cases of the language can be devised. Therefore, they make the provision of including a user updated lexicon. The user is able to update the standard system with any new words or pronunciations not originally included. In the case of new unrestricted text, however, such a solution can work only if someone has already read the text segment and "tuned" the user lexicon accordingly. The limitations of such a method are quite obvious and are a major factor in the less than enthusiastic acceptance of speech synthesizers in the computer market.

CH2674-5/89/0000-0765$01.00©1989IEEE

Neural Nets for Phonemic Transcription

Since the beginning of this decade, renewed interest in parallel distributed processing (PDP) has resulted in the development of several neural network models and learning algorithms [6]. The application areas are expanding and include a variety of problem areas from optimization applications to image and speech processing. Our knowledge of cognitive operations is still in its infancy, and the details of the activities involved there are very sketchy. What makes neural networks appealing to modeling cognitive activities is their principles of operation which are based on what we know about human neural networks, even though this knowledge is very limited. Also appealing is the fact that neural networks incorporate knowledge and memory, not stored in a separate static region and searched when the need arises, but inherently in the neural synaptic connections themselves. In short, the artificial neural knowledge, similar to the human case, is implicitly present, rather than explicitly represented.

Reading aloud from unrestricted text written in a particular language is a typical human cognitive activity, which indicates that a person has achieved literacy. Possessing this "skill" usually means that a person has some minimum level of "intelligence" and that during the early stages of life has undergone relatively long training on how to read. In essence, the reading operation involves the recognition of text, the retrieval from memory of the previous knowledge related to the nature of the text, and the control of the voice production system so that the output sound will be in agreement with some important characteristics which make speech intelligible and natural.

The first neural network model to be developed and used explicitly for speech synthesis is the NETtalk [7]. During the learning phase this network extracts the rules necessary for mapping letters into phonemes. It can be used as the front-end of a speech synthesizer, bypassing its high level analysis stages and presenting strings of phonemes for synthesis. Although many issues still remain unresolved, the relative success of the above system indicates that there are a number of potential applications where neural networks can be used in the speech synthesis area.

Problem Description

The contribution of NETtalk in introducing neural nets in speech synthesis has been important. However, a number of problems still remain to be solved. In this paper a subset of the speech synthesis problem is investigated, using a new approach. Instead of performing full phonemic transcription, as is the case with NETtalk, the present goal is to locate the letters which contribute to the phonology indirectly by affecting neighboring phonemes, while they themselves remain silent. Examples include the

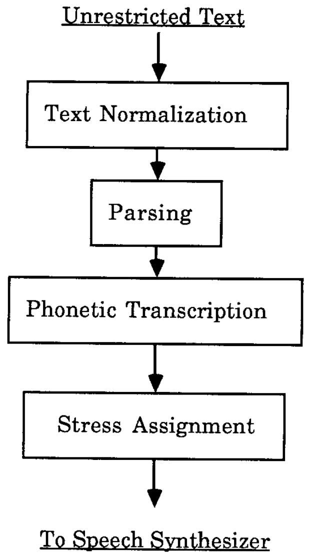

Figure 1. Sections of High Level Processing in a Speech Synthesis System.

words: somE, wHicH, tHus, difFerencE, etc., where the letters shown in capital can be considered as modifiers. It is important though, that any changes performed on the incoming text do not alter the phonetic information and the context. This is achieved by retaining the original text form, but indicating which letters in the incoming sequence act as modifiers. In the problem under considera- tion, the involved letters are viewed as carrying information implicitly by affecting neighboring letters.

A scheme is desired which will resolve the multiple-letter-to-one-letter correspondence problem before the text is presented to the synthesizer network. The proposed scheme should work on the following principles:

1. One-letter-one-sound correspondence:

No particular action will be taken, and the input text will pass unchanged.

2. Two or more letters to one-sound correspondence:

i. Double consonants (liTTle, diFFerent, alL, aCCent).
ii. Letters with no sound (littlE, differencE, neiGHbor).
iii. Letters which work as modifiers (acE, ratE, THis, abovE).

In these cases the letters involved should be marked to indicate that they belong to one of the above categories, and fed to the main network. Whenever the network is presented with such marked letters, it will pass to the next letter in the text. The effect of the marked letters will be indirect; they will affect the context, and therefore the pronunciation, of the involved words.

The Preprocessing Network Architecture

The proposed network is similar to that proposed by Sejnowski and Rosenberg for NETtalk, the phonetic transcription network. Its architecture is described as follows:

The Input Layer

Input is unlimited text, and as such, consists of combinations of all 26 letters of the alphabet, the space bar, comma, period and question mark, for a total of 30 symbols. All other symbols used in writing are considered to have little effect on the spoken text, and in any case their possible effect can be resolved in a conventional text preprocessing stage. The total of 30 symbols can be expressed by 5 binary bits of information. Each letter and symbol is assigned a number from 0 to 29 in a way that groups them according to their phonetic properties. Therefore, the vowels are grouped together, and the consonants are subdivided into categories according to their acoustic properties [8]. Grouping the input symbols in a logical, rather than random, way is known to increase the learning rate and efficiency of parallel distributed networks [9]. The context is captured by considering each letter as part of a 9-letter window, so that the possible effects of 4 letters on either side of the central, 5th letter under focus will be present. The text will slide in a 9-letter window one letter at a time. The 30 possible input symbols can be expressed by 5 binary bits of information. A total, then, of 45 neurons will comprise the input layer. The activation function of the input neurons is linear, and as such it passes the input signal unchanged.

The Output Layer

As mentioned above, whenever one-letter-one-sound correspondence is present, no action is required. For all other cases, the letter(s) involved should be marked. This scheme requires a simple binary decision, and for the output layer one neuron alone will suffice. Whenever this neuron is off, it will mean no action; when on, it will mean that the letter involved should be marked. This marking can be done either by placing an asterisk in front of the letter or by capitalizing it. The particular marking method is not important. The activation function of the output neuron is a sigmoid function [6]

$$\phi(x) = \frac{\text{max-min}}{1+e^{-x}} + \text{min},$$

and is shown in figure 2 below. The minimum and maximum values of the sigmoid were 0 and 1, respectively. Because the output can assume only on or off states, all output values greater or equal to 0.6 were considered as on and all less than or equal to 0.4 as off. The interval between 0.4 and 0.6 is a "no man's land."

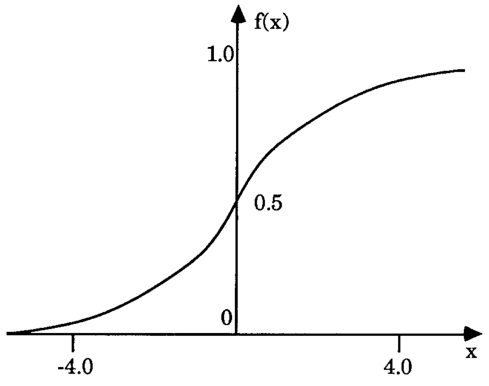

Figure 2. The Sigmoid Activation Function.

The Hidden Layer

A two layer neural network has been shown to be able to solve problems that involve mapping with a linearly separable multidimensional function[10]. For the solution of the problem at hand, it is most likely that the required function is highly non-linear. It is this property which makes the use of an extra hidden layer necessary [6]. The actual number of neurons cannot be exactly predefined, since no theory exists for such a task, particularly for cognitive problems. Experiments have been conducted with 10, 20, 30 and 40 neurons, to determine the effects of the size of the hidden layer on the learning and success rates. The activation function of the hidden layer is also a sigmoid.

The Network Connectivity

In this network only feed-forward connections are necessary. Therefore, the input layer feeds the hidden layer, which in turn feeds the output layer. The network architecture is shown on figure 3 below. L_n denotes the letters present at the input layer. Each letter is mapped into a 5-bit code as described above. The W_m are matrices consisting of the values of the synaptic interconnections.

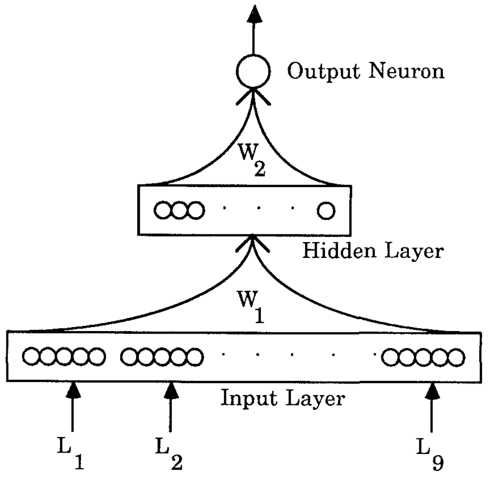

Figure 3. The Text Preprocessing Network.

Simulation and Results

The training corpus chosen for these experiments consisted of the 1100 most common words of English [11]. Surprisingly, these words are of the most difficult in terms of peculiarities in their pronunciation. Before training commenced the network weights were initialized with uniformly distributed random real values between -0.5 and 0.5. For the training stage all letters involved were hand-marked according to the scheme described above. The words of the training corpus were then fed into the network randomly. The learning method for this supervised learning problem is the Back Propagation algorithm [6]. This learning method requires that the desired response also be known during training, since it is used in the learning algorithm. The original Back Propagation algorithm was slightly modified with the addition of a small constant to the derivative of the activation function, so that chances of the network getting trapped in local minima are reduced [12]. During training the error rate was monitored and showed an almost monotonically decreasing behavior. Results for 100, 500 and 1100 words were obtained for networks with hidden layers of sizes 30 and 40 neurons and are shown on figures 4 and 5, respectively. For the training corpora of 100 words the network converged to 100% success rate after 86 iterations for a hidden layer of size 30, and after 75 iterations for a hidden layer of size 40. Because the simulation was carried out on a microcomputer, the training corpora of size 500 and 1100 words proved to be excessively large, and the error only asymptotically approaches 0. However, convergence is expected to occur after a sufficient number of iterations.

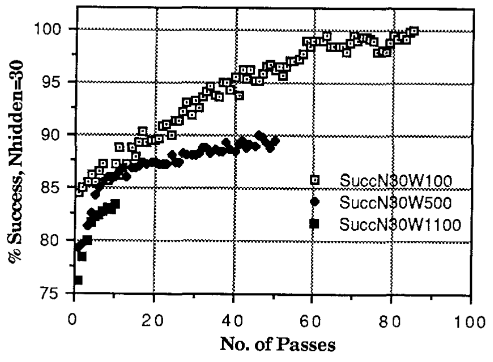

Figure 4. Success Rate for 30 Hidden Neurons.

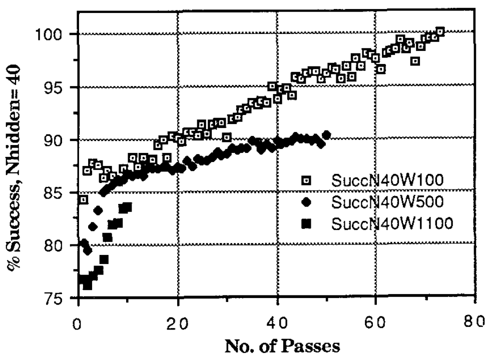

Figure 5. Success Rate for 40 Hidden Neurons.

Observations and Conclusions

The results obtained from simulating the neural networks showed that even a relatively small network has the ability to extract the rules inherent in a mapping of a problem with the complexity described here. The experiments run were a compromise between the network size and the time required for convergence. The results show that the larger the network the faster the learning rate but also greater the computational requirements. Problems of such a magnitude would preferably be processed either on a larger system or on a dedicated processor. The network derived from the training will be used as the front end of a new speech synthesis method currently under development. Improvements on the algorithm are being considered.

References

[1] Lazzaro, J. J., "Voice Synthesis," *CD-ROM Review,* Vol. 3, No.8, October 1988.

[2] Klatt, D., "Review of text-to-speech conversion for English," *J. Acoust. Soc. Am.* 82(3) Sept 1987, pp. 737-793.

[3] Umeda, N., "Linguistic Rules fro Text-to-Speech Synthesis," *Proceedings of the IEEE,* Vol. 64, No. 4, April 1976.

[4] Allen, J., "Synthesis of Speech from Unrestricted Text, "*Proceedings of the IEEE,* Vol. 64, No. 4, April 1976.

[5] Allen, J., Hunnicutt, M. S. and Klatt, D., *From text to speech: The MITalk system,* Cambridge University Press, Cambridge, 1987.

[6] Rumelhart, D. E., Hinton, G. E. and Williams, R. J. "Learning Internal Representations by Error Propagation", in *Parallel Distributed Processing, Explorations in the Microstructure of Cognition, Volume 1: Foundations.* The MIT Press, Cambridge Massachusetts, 1986.

[7] Sejnowski, T. J. and Rosenberg, C. R., *NETtalk: A Parallel Network that Learns to Read Aloud.* The John Hopkins University Electrical Engineering and Computer Science Technical Report JHU/EECS-86/01, 1986.

[8] Rabiner, L. R. and Schafer, R. W. *Digital Processing of Speech Signals,* Prentice-Hall, Englewood Cliffs, New Jersey, 1978.

[9] Jordan, M., *Serial Order: A Parallel Distributed Approach.* Institute for Cognitive Science University of California, San Diego, ICS Report 8604, May 1986.

[10] Minsky, M and Papert, S., *Perceptrons,* The MIT Press, Cambridge, Massachusetts, 1969.

[11] Kucera, H. and Francis, W. N., *Computational Analysis of Present-Day American English,* Brown University Press, Providence, RI, 1967.

[12] Fahlman, S. E., *An Empirical Study of Learning Speed in Back- Propagation Networks,* CMU-CS-88-162, Carnegie Mellon University, 1988.

Michael S. Scordilis was born in Corfu, Greece. He received the B.E. in communications engineering from the Royal Melbourne Institute of Technology, Melbourne, Australia, and the M.S. in electrical engineering from Clemson University in 1984 and 1986, respectively. He is currently a Ph.D. candidate in Electrical Engineering at Clemson University.

His research interests include speech synthesis and compression and parallel distributed modeling, as applied to speech processing. He is a student member of the IEEE Computer and Acoustics Speech, and Signal Processing Societies.