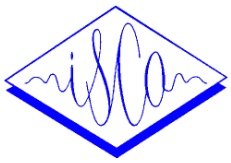

The ESCA Workshop
on Speech Synthesis
Autrans, France
September 25-28, 1990

A MULTI-LANGUAGE TEXT-TO-SPEECH SYSTEM USING NEURAL NETWORKS

Tatsuro Matsumoto, Yukiko Yamaguchi

Software Laboratory
Fujitsu Laboratories Ltd.
1015 Kamikodanaka, Nakahara-ku, Kawasaki 211, JAPAN

ABSTRACT

In this paper, the design philosophies and performances of two components of our multi-language text-to-speech system are presented. A syntactic boundary neural network is trained with many five-word sequences and used to determine the boundaries existing before a middle word within a given word sequence. A letter-to-phoneme conversion neural network converts input letters to phonemes. To ensure reliability, we employed multiple networks and a unification layer. Results of performance evaluation for English show that the syntactic boundary neural network correctly located the syntactic boundaries with 96% accuracy (trained with 500 sentences, and tested with another 500 sentences), and that the letter-to-phoneme conversion neural network correctly converted letters to phonemes with 85% accuracy (trained with 1000 words, and tested with another 1000 words).

1. INTRODUCTION

The research team at Fujitsu Laboratories is currently developing a multi-language text-to-speech system. Since most of the conventional text-to-speech systems [1] are language specific and rule-based, expanding them to handle multiple language is often inefficient and costly. The neural network approach provides an alternative in which language specific rules are automatically extracted in the form of weight sets.

Our text-to-speech system is shown in Figure 1. This system extracts syntactic boundaries from an input part-of-speech sequence and converts letters into phonemes using neural networks realized by a software simulation on a UNIX system. In this paper, we will describe our Syntactic Boundary Neural Network, and Letter-to-Phoneme Conversion Neural Network.

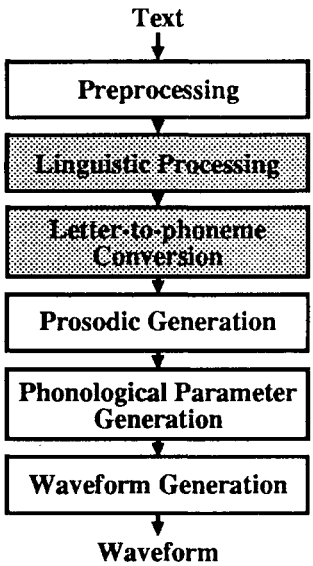

Figure 1 Processing Flow of the Text-to-Speech Conversion System

2. SYNTACTIC BOUNDARY NEURAL NETWORK

To generate natural prosodic patterns, we employed Fujisaki's model [2] as the fundamental frequency pattern model. To control this model, it is necessary to extract syntactic information such as part-of-speech, syntactic boundary etc. Using conventional methods to extract syntactic

information from complex and/or ill-formed sentences requires a large system, and even then, the information cannot be extracted perfectly. We developed a method which extracts phrases from part-of-speech sequences using a neural-network. This network outputs whether syntactic boundaries exist before a middle word (the word which is located in the center) of the sequences.

2.1. Structure

The structure of the network is shown in Figure 2. The network consists of 3 layers. The input layer has units corresponding to the part-of-speech for each word. The number of units equals C*N, where C is the number of categories of part-of-speech, and N is the number of input words (in this paper, C and N are set to 20 and 5 respectively). The output layer has units corresponding to the classes of syntactic boundaries located before the middle word of the input. The classes of syntactic boundaries are noun phrase, verb phrase, preposition phrase, infinite phrase boundary and clause boundary. If there exist boundaries before the middle word, the units for those boundaries are fired. In Figure 2, for example, when a part of sentence, "so nice to sit here", is presented, the units for the input part-of-speech are fired at the input layer, and the unit for an infinite phrase boundary is fired at the output layer since there is an infinite phrase boundary before the middle word "to."

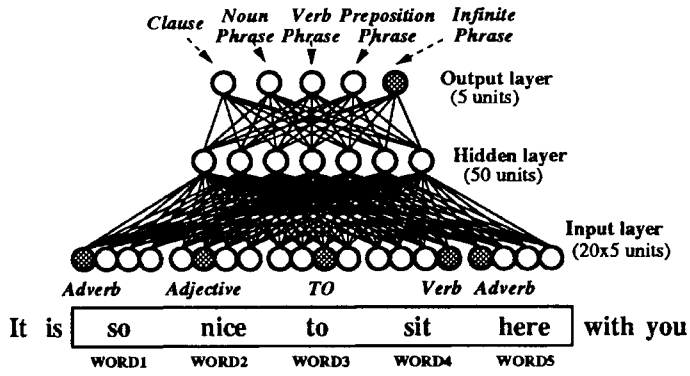

Figure 2 The Structure of the Phrase/Clause Boundary Neural Network

2.2. Training

We divided 956 English sentences into 2 sets; one is a training set, and the other is an evaluation set. The number of part-of-speech categories is 20. The network was trained by 4073 pattern 100 times/pattern using back propagation [3].

2.3. Discussion

Table 1 shows the rate of locating boundaries correctly. The rate of location of phrase boundary was more than 95% in the closed and open test. For clause boundaries, however, the rate is only 67.2% in the open test. The reason for this low rate is that the number of training patterns for the clause boundaries is less than for the phrases boundaries. In this case the number of training patterns for noun and verb phrases are 973 and 700 respectively, whereas, the number of training patterns for clause boundaries is 124.
The number of training patterns for preposition and infinite boundaries are as small as for the clause boundaries, but since the part-of-speech of the function word as the cue of the boundaries is obvious, it is easy for the network to extract the boundaries. Therefore to improve the rate of location of clause boundary, the network has to be trained by set of sentences which include as many clause boundaries as noun or verb phrases.

Table 1 The rate of location of correct boundaries for each boundary

Class of Boundaries	Closed Test		Open Test	
	Correct Boundaries Rate (%)	Number of Boundaries	Correct Boundaries Rate (%)	Number of Boundaries
Clause	90.3	124	67.2	122
Noun phrase	99.4	973	95.4	969
Verb phrase	99.6	700	95.9	702
Preposition phrase	100.0	151	97.7	216
Infinite phrase	100.0	124	97.8	134
No boundary	98.6	2094	93.8	2083
Total	98.3	4073	95.0	4091

3. LETTER-TO-PHONEME CONVERSION NEURAL NETWORK

Several rule-based systems [4] have been developed for converting input texts to phoneme symbols which describe how to pronounce the spelling. To develop the rule sets in these systems takes a long time and requires a knowledge of the target language. We tried to make neural networks learn letter-to-phoneme conversion rules automatically, like NETtalk [5], using multiple neural networks and training each network by various contexts.

3.1. Structure

A letter-to-phoneme conversion neural network is shown in Figure 4. This network consists of multiple networks and a unification layer. Each network has 3 layers as in NETtalk (Figure 3). Each network outputs the distinctive feature of the phoneme of the spelling located at the position specified for each network. The unification layer integrates the outputs of each network, and outputs the distinctive feature of the phoneme. The weight of the link between the output units of each network and each unit of unification layer is 1/N, where N is the number of networks.

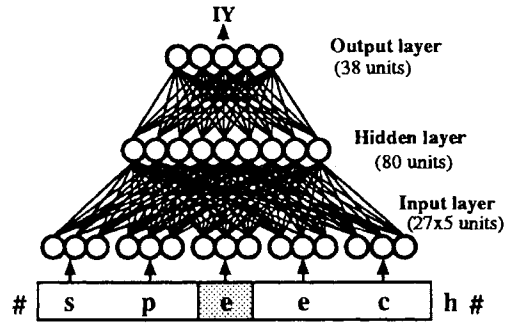

Figure 3 The Structure of the Element of the Letter-to-Phoneme Conversion Neural Network

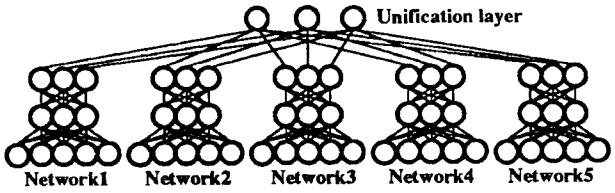

Figure 4 The Structure of the Letter-to-Phoneme Conversion Neural Network

Network1		Network2		Network3		Network4		Network5	
Input spellings	Phoneme	Input spellings	Phoneme	Input spellings	Phoneme	Input spellings	Phoneme	Input spellings	Phoneme
speec	S	#spee	S	##spe	S	###sp	S	####s	S
peech	P	speec	P	#spee	P	##spe	P	###sp	P
eech#	IY	peech	IY	speec	IY	#spee	IY	##spe	IY
ech##	-	eech#	-	peech	-	speec	-	#spee	-
ch###	CH	ech##	CH	eech#	CH	peech	CH	speec	CH
h####	-	ch###	-	ech##	-	eech#	-	peech	-

Figure 5 Examples of input spellings and teacher phonemes for each network

3.2. Training

Each network is trained with five spelling sequences and the distinctive feature of the phoneme at the specified position (Figure 5). In this figure, the spelling sequence "speec" is provided and phoneme [s] as the pronunciation of the first spelling 's' is learned by the first network. The sequence "peech" is then provided and phoneme [p] as the pronunciation of spelling 'p' is learned. For the second network, "#spee" is provided with '#' representing the word boundary and [s] as the pronunciation of the second spelling 's' is learned. Then "speec" is provided and phoneme [p] as the pronunciation of spelling 'p' is learned. In the same way, each network is trained with different contexts.

We divided 2000 English words into 2 sets; one is a training set, and the other is an evaluation set. For all spellings of the training words, each network is trained until the outputs have less errors than a given threshold. The weights of the networks are modified by back propagation.

3.3. Discussion

In Table 2, the performance of the letter-to-phoneme conversion neural network consists of a single network, 3 networks and 5 networks are shown. In this table, the multiple-network shows a higher rate of phoneme and word conversion than the single-network. Table 3 shows the conversion rate of each phoneme and stress. For consonants, the closed and open test show more than 94%. For vowels, however, the closed test shows 89.7%, and the open test shows 68.2%. Because there are less contexts affecting pronunciation of consonant, the rules for

consonants could be learned by these training words. However, it seems that pronunciation of a vowel varies depending on the context, since the rule for a vowel could not be derived from this training word set. And since this training word set includes words which have a pronunciation exception, the network cannot learn by these words.

For stress, primary stress and unstressed show 90.5% in the open test. But no secondary stress is converted correctly. Because the number of training patterns for primary stress and unstressed are 1024 and 4561, while the number of training patterns for secondary stress are only 57 patterns, the secondary stresses are not learned. Primary stresses are learned by a shorter context, while secondary stresses are learned by a longer context including the primary stress. Thus, in this case secondary stresses are not learned by a 5-letter context in each network. In order to convert correct stress, it is necessary to train the networks identify secondary stress with primary stress and decide the strength of the stress in other method.

In Table 2, the word conversion rate of multiple-networks is 10% higher than that of single-network. However, the word conversion rate is much less than the phoneme conversion rate. It seems that this is caused by a low rate of vowel conversion. To convert a word into correct phonemes, the pronunciation of vowels should be learned sufficiently. Therefore, it is necessary to train the networks using a large vocabulary without any exception words.

Table 2 The rate of conversion of correct phonemes and words

Number of Networks	Closed Test		Open Test	
	Correct Phoneme Rate (%)	Correct Word Rate (%)	Correct Phoneme Rate (%)	Correct Word Rate (%)
1	88.1	51.2	80.8	34.1
3	92.2	63.9	83.4	44.2
5	94.0	72.8	84.6	46.3

Table 3 Table of correct conversion rate for each class of phonemes and stresses

Class of phonemes /stresses	Closed Test		Open Test	
	Correct Rate (%)	Number of Phonemes	Correct Rate (%)	Number of Phonemes
Vowel	89.7	1663	68.2	1699
Consonant	98.9	3979	94.3	3951
Primary Stress	98.6	1024	90.5	1019
Second Stress	0.0	57	0.0	63
Unstressed	98.8	4561	94.6	5650

4. CONCLUSION

In our evaluation, both of the syntactic boundary neural network and letter-to-phoneme conversion neural network show good performances. In the future we are going to evaluate the performance of the networks by using plenty of sentences and large vocabulary, and integrate the networks into our text-to-speech system. As the next step, we are also going to develop a text-to-speech system for other languages to establish the validity of these networks.

REFERENCES

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

[2] Fujisaki, H., Hirose, K. & Kawai, H. (1985) "A System for Synthesis of Connected Speech - Special Emphasis on the Synthesis of Prosodic Features - ", Trans. of Committee on Speech Research, Acoust. Soc. Japan S85-43

[3] Rumelhart, D.E., Hinton, G.E. & Williams R.J. (1986) "Learning representations by back-propagating errors", NATURE Vol.323 9 Oct.

[4] Allen, J., Hunnicutt, M.S. & Klatt, D.H. (1987) "From text to speech: The MITalk system", Cambridge Studies in Speech Science and Communication, Cambridge University Press.

[5] Sejnowski, T.J. & Rosenberg, C.R. (1986) "NETtalk: A Parallel Network that learns to Read Aloud", The Johns Hopkins University Electrical Engineering and Computer Science Technical Report JHU/EECS-86/01