APPLICATION OF THE NEURAL NETWORKS FOR TEXT-TO-PHONEME MAPPING

Eniko Beatrice Bilcu[1], *Petri Salmela*[2], *Janne Suontausta*[3], *Jukka Saarinen*[4]
[1,2,4]Digital and Computer Systems Laboratory, Tampere University of Technology,
Korkeakoulunkatu 1, FIN-33720, Tampere, Finland
[3]Nokia Research Center, Speech and Audio Systems Laboratory,
P.O. Box 100, FIN-33721, Tampere, Finland
`e-mail:` [1]`bilcub@cs.tut.fi`

ABSTRACT

In this paper we present the results on the use of neural networks for text-to-phoneme mapping. For this mapping, we have compared the performances of the Context Dependent Multilayer Perceptron network with the Recurrent Neural Network. The results (number of parameters vs neural network model vs phoneme accuracy) are given for American English. Also some guidelines for selecting the appropriate network structure together with some methods for improving the phoneme accuracy are presented.

Keywords: Neural Networks, Multilayer Perceptron Network, Recurrent Neural Network, Text-to-Phoneme Mapping.

1 INTRODUCTION

Speech recognition has been one of the most important research topic in human-computer interaction. Our paper deals with a narrow area of human computer interaction, namely the text-to-phoneme (TTP) mapping [3]. In speaker independent speech recognition systems based on phonemes, there is a necessity to define a dictionary in terms of phonetic transcriptions. The transcriptions can be based on a predefined look-up table or some mathematical model that gives the transcription for any given word. Clear, the benefit of the previous method is that it is more accurate than the latter one. However, the model based approach requires less memory and can be straightforwardly applied for new words.

Neural networks (NNs) has been shown to provide a solution for this problem. In [4], MLP networks are applied to the TTP mapping, the mapping is based on the context information in which the letters occur. However, in that paper there is no comparative study on the use of different network structures. For real time applications, the requirement is to obtain fast networks that uses a small amount of memory and also can provide accurate recognition rates. Usually, each letter and each phoneme are represented as a binary vectors for the neural network. If these vectors has large dimensions, also the input-output layers of the NN scales correspondingly, that easily results in large number of weights especially if context vectors are used. Although the context dependency has been shown to increase the phoneme recognition accuracy, the large number of the parameters of such NN is not desired from the application point of view. Based on these well known observations we have studied the recognition results obtained by NNs with simple structures, such as Recurrent Neural Networks (RNNs) compared with the context dependent MLP approaches.

2 DATABASE

The dictionary used for training and testing the neural networks in our experiments was the Carnegie Mellon University (CMU) pronunciation dictionary. In order to implement TTP with NNs, the data from the dictionary has to be preprocessed in the following manner:

- The words and their phoneme transcriptions were aligned such that one-to-one correspondence was obtained between the letters of each word and its phoneme symbols. The alignment was done using a Viterbi algorithm [7].

- In order to eliminate the ambiguity that can occur for multiple pronunciations of the same word, only one phonetic transcription was chosen from each entry into the dictionary.

- The whole dictionary was splitted into two parts. For the first part we have randomly chosen 80% from the whole CMU dictionary (each word with a single phonetic transcription). The training dictionary used in our simulations contains the phonetic transcriptions of 86821 words. The number of patterns (input vectors) in the input training set was around 65×10^4. The second part contained the rest 20% (22015 words) of the whole CMU dictionary and it is used for testing the phoneme accuracy of the models. The set used for training the NNs, and the set used for testing the NNs did not contain words in common.

- Once we have obtained the training and test set, both sets are processed as follows: First, the order of the words in both sets were randomized. After that, each letter in a word is coded using binary vectors such as shown in Table 1, where \0 is introduced to represent the *graphemic null*. The number of letters in the English dictionary is 26, and together with a *graphemic null* we have 27 letters. Therefore, each vector, representing a letter in a word, or space between words, have 27 elements. Similar coding scheme was also applied for the phoneme transcriptions. Since English can be represented with 47 phonemes including the *null phoneme* and *pseudo phonemes*, the dimension of the binary vector that codes the phoneme is 47 such as shown in Table 2.

Letters	Corresponding binary vector
a	1 0 0 0 ... 0 0 0
b	0 1 0 0 ... 0 0 0
⋮	⋮
y	0 0 0 0 ... 1 0 0
z	0 0 0 0 ... 0 1 0
\0	0 0 0 0 ... 0 0 1

Table 1: Orthogonal letter codes. Each vector has 27 elements of which only one is set to value of one.

Phonemes	Corresponding binary vector
-	1 0 0 0 ... 0 0 0
aa	0 1 0 0 ... 0 0 0
⋮	⋮
zh	0 0 0 0 ... 0 0 1

Table 2: Orthogonal phoneme codes. Each vector has 47 elements of which only one is set to value of one.

3 NEURAL NETWORK ARCHITECTURES

The neural network structures compared in our paper were the Multilayer Perceptron (MLP) network and the Recurrent Neural Network (RNN).

3.1 Multilayer Perceptron (MLP)

For the MLP, in order to take into account the grapheme context, a number of letters on each side of the current letter were also used as input to the network. We have tested the MLP having as input three adjacent letters and five adjacent letters with the middle one being the letter to be transcribed. Also in our paper we present the phoneme accuracy obtained with MLPs that used the letters just from the left context. These results are included just for benchmark purposes.

In Figures 1 and 2, the two structures of the multilayer perceptrons used in the simulations are shown. In Figure 1, the input vector $u(n)$ consists of three adjacent letters, whereas in the second implementation shown in Figure 2 the input vector is obtained by concatenation of five vectors corresponding to five adjacent letters. Both MLPs have one input, hidden and output layer $y(n)$ and both of these networks are fully connected. This means that each neuron in the hidden layer receives inputs from each unit in the input layer. Furthermore, each hidden unit sends its output to all the units of the output layer.

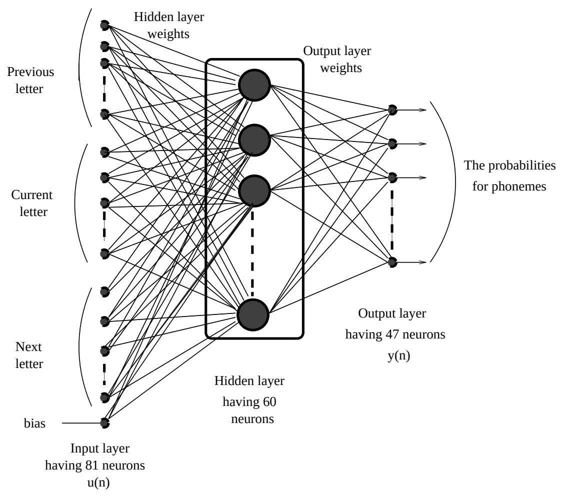
Hidden layer weights
Output layer weights
Previous letter
Current letter
Next letter
bias
The probabilities for phonemes
Output layer having 47 neurons y(n)
Hidden layer having 60 neurons
Input layer having 81 neurons u(n)

Figure 1: Multilayer Perceptron with 3 input letters, which are represented with the binary vectors shown in Table 1.

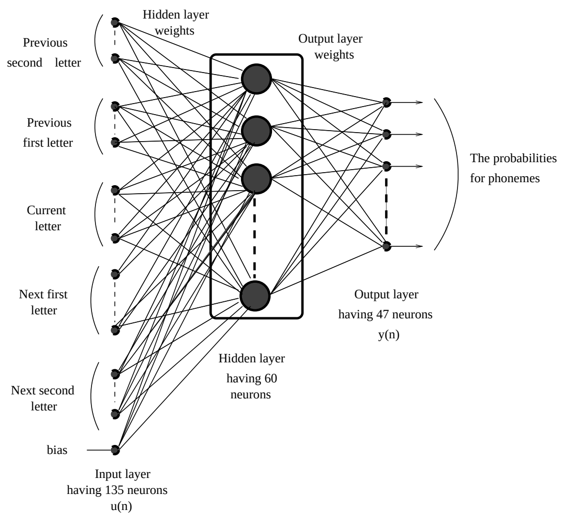

Figure 2: Multilayer Perceptron with 5 input letters, which are represented with the binary vectors shown in Table 1.

For the hidden layer, we have used the hyperbolic tangent activation function which can be given as:

$$x_i = \frac{1 - exp(av_i)}{1 + exp(av_i)} \tag{1}$$

where a is a constant, v_i is the induced local field of the i^{th} neuron and x_i is the output from the i^{th} hidden neuron.

The activation function of the output layer was chosen to be the softmax function:

$$P_i = \frac{exp(bv_i)}{\sum_{i=1}^{N} exp(bv_i)} \tag{2}$$

where b is a constant, N is the number of outputs and P_i is the i^{th} output.

The softmax activation function give a good approximation of the class posterior probabilities [2, 4]. Therefore in the testing procedure when one pattern (one letter or a group of letters) is presented at the input of the network, at the output we obtain a vector with 47 elements that has a maximum value in the position corresponding to the recognized phoneme. The recognized phoneme is then finded using the following criterion:

$$C_c = \underset{i}{argmax} \quad (P_i)$$

where C_c is the index of the recognized phoneme from the list of total phonemes (see Table 2).

3.2 Recurrent Neural Network (RNN)

For the RNN there is no necessity to introduce context dependency into the input vector since the output state is feedback to the input. The context dependency is incorporated into the feedback. The structure of the fully connected RNN used in our experiments is depicted in Figure 3. The input vector $u(n)$ is formed by the binary vector of the current letter and $y(n)$ is the binary vector containing probabilities estimated for the phonemes. All outputs are feedback to the input and they represent the network state. The first 47 states are taken as the outputs of the network and represents the corresponding phoneme. The activation function used in this case was the softmax.

4 TRAINING THE NETWORKS

The training of the RNN was based on the truncated back-propagation through time algorithm (BPTT). The training procedure may be derived by unfolding the temporal operation of the network into a layered feed forward network. In order to maintain a feasible computational complexity, the relevant history of input data and network states are saved for a fixed number of time steps. A detailed description of the training algorithm can be found in [1, 2]. The number of time steps for which the data is stored is called the truncation depth

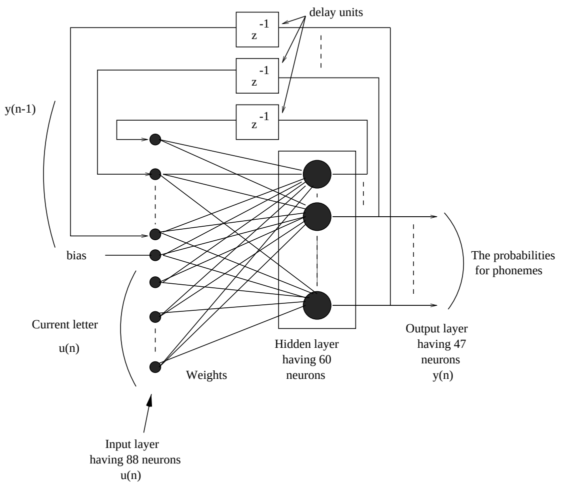

Figure 3: Fully connected Recurrent Neural Network.

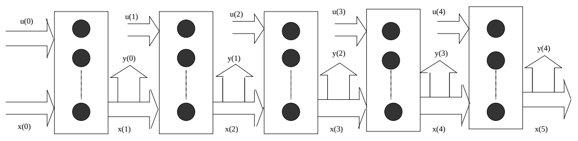

Figure 4: Recurrent Neural Network unfolded in time for a truncation depth of 5 time steps.

[1, 2, 6]. We have implemented two RNNs, one with a truncation depth of 3 time steps and the other one with a truncation depth of 5 time steps (shown in Figure 4). The MLPs tested in our paper were trained using the back-propagation algorithm.

Because the training dictionary contains a large amount of data, in order to have a faster training, all the networks are trained in on-line mode and the weights are updated after presentation of each training example.

The complexity of the compared neural networks are given in Table 3, where by MLP1 we have denoted the Multilayer Perceptron having just one left-side context dependence, MLP2 is the Multilayer Perceptron with two left dependences, MLP3 is the Multilayer Perceptron with one letter on both sides of the current letter, MLP5 is the Multilayer Perceptron with two letters on both sides of the current letter, RNN3 is the Recurrent Neural Network trained with a truncation depth of 3 time steps and RNN5 is the Recurrent Neural Network trained with a truncation depth of 5 time steps. Note that each compared network contains also the bias terms. As we can see from Table 3, the RNN models needs less memory to store the final weights than the other implementations.

5 EXPERIMENTAL RESULTS

In order to test the recognition performances of each neural network a large number of simulations were performed. During the simulations different parameters settings were tested. The synaptic weights were initialized with values chosen from an uniform random distribution between -0.02 and 0.02. All the networks were trained using a constant learning rate $\lambda = 0.01$. The constants a and b for the activation functions were chosen equal with 5. In order to have a fair comparison the number of hidden neurons in the MLPs and the number of state neurons in the RNNs were chosen the same $N = 60$. During the training, some intermediate values of the synaptic weights were saved and the testing procedure is performed using these values. The synaptic weights were saved at 1%, 2%,...,100% from the whole training set. In Figure 5 the phoneme accuracy of test set is plotted as a function of this percentage. Figure 5 shown that the higher phoneme accuracy are obtained with MLP3 and MLP5. This was expected since the MLP3 and MLP5 incorporates context dependences from both sides of current letter. Approximately the same phoneme accuracy were obtained for MLP1, MLP2, RNN3 and RNN5. This is because all these networks incorporates context dependence just on the left side of the current letter. An interesting result can be observed in Figure 5 comparing the phoneme accuracy for RNN3 and RNN5. Since the phoneme accuracies, were the same for RNN3 and RNN5, we can conclude that in the application addressed here there is no necessity to train a RNN model with a truncation depth of more than 3 time steps. Therefore, using a RNN with a small truncation depth we can obtain the same phoneme accuracy but with considerably less computational effort. Also, comparing the sizes of the networks given in Table 3 and the recognition results shown in Figure 5 we can conclude that the MLPs with both sides context dependencies performs better than the RNN, but they need a large number of weights.

	Input Length	Output Length	Hidden Neurons	Syn. Weights
MLP1	54	47	60	6167
MLP2	81	47	60	7787
MLP3	81	47	60	7787
MLP5	135	47	60	11027
RNN3	27	47	60	5280
RNN5	27	47	60	5280

Table 3: The size of Neural Networks.

6 CONCLUSIONS

In this paper, MLP network and RNN architectures are tested for the problem of text-to-phoneme mapping. According to results we can conclude that the RNN provides smaller phoneme accuracy than the MLP with context dependencies from both sides. However, when MLP have only the left context, the phoneme accuracy

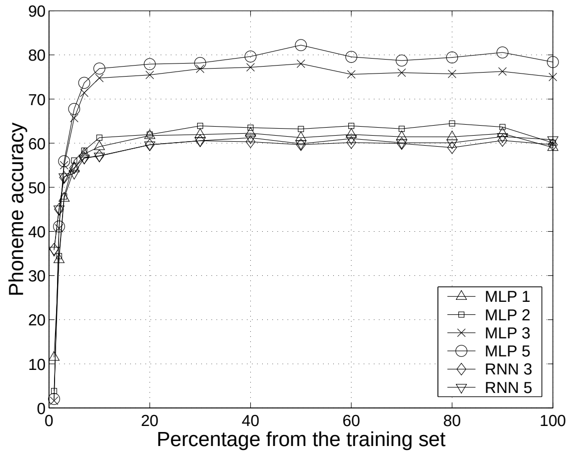

Figure 5: Phoneme accuracy for the tested NNs.

of MLP and RNN are approximately same. Furthermore, the memory needed for storing the weights is much smaller in the case of RNNs. Also, we have observed in our case, that the training of the RNN can be performed by using a small truncation depth without loss in the phoneme recognition accuracy. The results also indicate that inclusion of also right-side context dependency into the RNN could lead to improved performance.

References

1. S. Haykin, *Neural Networks - A Comprehensive Foundation*, 2nd Ed., Pretince Hall, New York, 1999.

2. C. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, 1995.

3. N. McCulloch, M. Bedworth and J. Bridle, *NetSpeak - a re-implementation of NetTalk*, Computer Speech and Language 2, 1987, p. 289-301.

4. K. Jensen and S. Riis, *"Self-Organizing Letter Code-Book for Text-to-Phoneme Neural Network Model"*, Proceedings of the International Conference on Spoken Language Processing, 2000.

5. M. Embrechts and F. Arciniegas, *"Neural Networks for Text-to-Speech Phoneme Recognition"*, Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Vol. 5, 2000, pp. 3582-3587.

6. M. Adamson and R. Damper, *"A Recurrent Network that Learns to Pronounce English Text"*, Proceedings of the International Conference on Spoken Language Processing, Vol. 3, 1996, pp. 1704-1707.

7. V. Pagel, K. Lenzo and A. Black, *"Letter to Sound Rules for Accented Lexicon Compression"*, Proceedings of the International Conference on Spoken Language Processing, Vol. 5, 1998, pp. 2015-2018.